

# Computer Vision

## Statistical Models for Marr's Paradigm

Song-Chun Zhu

Ying Nian Wu

April 22, 2022





# Contents

<b>Preface</b>	<b>ix</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Goal of Vision . . . . .	3
1.2 Seeing as Bayesian Inference: Vision is Ill-Posed and Mostly an Illusion . . . . .	6
1.3 Knowledge Representation . . . . .	7
1.4 Pursuit of Statistical Models . . . . .	12
<b>2 Statistics of Natural Images</b>	<b>17</b>
2.1 Image Space and Distribution . . . . .	18
2.2 Information and Encoding . . . . .	19
2.3 Image Statistics and Power Law . . . . .	21
2.4 Kurtosis and Sparsity . . . . .	26
2.5 Scale Invariance . . . . .	29
<b>3 Textures</b>	<b>33</b>
3.1 The Julesz Quest . . . . .	33
3.2 MRF & Clique-Based Gibbs Models . . . . .	35
3.2.1 Markov Random Fields (MRF) . . . . .	35
3.2.2 Ising & Potts Model . . . . .	36
3.2.3 Gaussian Markov Random Fields (GMRF) . . . . .	38
3.2.4 Advanced Models: Hierarchical MRF and Mumford-Shah Model . . . . .	41
3.2.5 Selecting Filters and Learning Potential Functions . . . . .	41
3.3 Filters for Early Vision . . . . .	41
3.3.1 Correlation & Convolution . . . . .	42
3.3.2 Edge Detection Filters . . . . .	43
3.3.3 Gaussian Filter . . . . .	44
3.3.4 Derivative of Gaussian & Laplacian of Gaussian Filter . . . . .	45
3.3.5 Gabor Filter . . . . .	46
3.4 The FRAME Model . . . . .	48
3.4.1 Intuition and the Big Picture . . . . .	48
3.4.2 Deriving the FRAME model . . . . .	51
3.5 The Texture Ensemble . . . . .	62
3.5.1 Ensembles in Statistical Physics . . . . .	62
3.5.2 Texture Ensemble . . . . .	64
3.5.3 Type Theory and Entropy Rate Functions . . . . .	65

3.5.4	Equivalence of FRAME and Julesz Ensemble . . . . .	69
3.6	Deriving Partial Differential Equations from the MRF and FRAME Models . . . . .	71
3.6.1	Turing Diffusion-Reaction . . . . .	72
3.6.2	Heat Diffusion . . . . .	72
3.6.3	Anisotropic Diffusion . . . . .	73
3.6.4	GRADE: Gibbs Reaction And Diffusion Equations . . . . .	73
3.6.5	Properties of GRADE . . . . .	76
3.7	Conclusion . . . . .	78
<b>4</b>	<b>Textons</b>	<b>79</b>
4.1	Distinguishing Textures and Textons . . . . .	79
4.1.1	Julesz's Confusion . . . . .	79
4.1.2	Neural Coding Schemes . . . . .	80
4.2	Sparse Coding . . . . .	81
4.2.1	Image Representation . . . . .	81
4.2.2	Basis and Frame . . . . .	82
4.2.3	Olshausen-Field Model . . . . .	83
4.2.4	A Three-Level Generative Model . . . . .	85
4.3	Active Basis Model . . . . .	90
4.3.1	Olshausen-Field Model for Sparse Coding . . . . .	90
4.3.2	Active Basis Model for Shared Sparse Coding of Aligned Image Patches . . . . .	91
4.3.3	Prototype Algorithm . . . . .	91
4.3.4	Statistical Modeling . . . . .	93
4.3.5	Shared Matching Pursuit . . . . .	95
4.3.6	Active Appearance Models . . . . .	97
4.4	The Sparse FRAME Model . . . . .	101
4.4.1	Dense FRAME . . . . .	103
4.4.2	Sparse Representation . . . . .	103
4.4.3	Maximum Likelihood Learning . . . . .	104
4.4.4	Generative Boosting . . . . .	104
4.4.5	Sparse Model . . . . .	106
4.5	Compositional Sparse Coding . . . . .	106
4.5.1	Sparsity and Composition . . . . .	106
4.5.2	Compositional Sparse Coding Model . . . . .	109
4.6	Bottom-up Filters or Top-down Basis Functions? . . . . .	112
<b>5</b>	<b>Gestalt Laws and Perceptual Organization</b>	<b>113</b>
5.1	Gestalt Laws for Perceptual Organization . . . . .	113
5.2	Texton Process Embedding Gestalt Laws . . . . .	114
5.2.1	Introduction . . . . .	114
5.2.2	Background on Descriptive and Generative Learning . . . . .	117
5.2.3	A Multi-layered Generative Model for Images . . . . .	119
5.2.4	A Descriptive Model of Texton Processes . . . . .	121
5.2.5	An Integrated Learning Framework . . . . .	125
5.2.6	Effective Inference by Simplified Likelihood . . . . .	128
5.2.7	Discussion . . . . .	132

<b>6</b>	<b>Primal Sketch: Integrating Textures and Textons</b>	<b>135</b>
6.1	Marr's Conjecture on Primal Sketch . . . . .	135
6.2	The Two Layer Model . . . . .	136
6.2.1	Structure Domain . . . . .	136
6.2.2	The Dictionary of Image Primitives . . . . .	138
6.2.3	Texture Domain . . . . .	140
6.2.4	Integrated Model . . . . .	140
6.2.5	The Sketch Pursuit Algorithm . . . . .	141
6.3	Hybrid Image Templates . . . . .	141
6.3.1	Representation . . . . .	142
6.3.2	Prototypes, $\epsilon$ - balls, and Saturation Function . . . . .	143
6.3.3	Projecting Image Patches to 1D Responses . . . . .	144
6.3.4	Template Pursuit by Information Projection . . . . .	146
6.3.5	Example: vector fields for human hair analysis and synthesis . . . . .	148
6.4	Relations between Primal Sketch and the HoG and SIFT Representations . . . . .	150
<b>7</b>	<b>2.1D Sketch and Layered Representation</b>	<b>155</b>
7.1	Problem Formulation . . . . .	156
7.2	The Variational Formulation by Nitzberg and Mumford . . . . .	157
7.2.1	The Energy Functional . . . . .	157
7.2.2	The Euler Elastica for Completing Occluded Curves . . . . .	158
7.3	The Mixed Markov Random Field Formulation . . . . .	158
7.3.1	Definition of $W_{2D}$ and $W_{2,1D}$ . . . . .	160
7.3.2	The mixed MRF and Its Graphical Representation . . . . .	162
7.3.3	Bayesian Formulation . . . . .	163
7.4	2.1D Sketch with Layered Regions and Curves . . . . .	164
7.4.1	Generative Models and Bayesian Formulation . . . . .	166
7.4.2	Bayesian Formulation for Probabilistic Inference . . . . .	169
7.4.3	Experiments . . . . .	171
<b>8</b>	<b>2.5D Sketch and Depth Maps</b>	<b>173</b>
8.1	Marr's Definition . . . . .	174
8.2	2.5D Sketch from Primal Sketch — Shape from Stereo . . . . .	175
8.2.1	The Image Formation Model . . . . .	177
8.2.2	Two Layer Representation . . . . .	181
8.2.3	The Inference Algorithm . . . . .	189
8.2.4	Example Results . . . . .	192
8.3	2.5D Sketch from Primal Sketch — Shape from Shading . . . . .	193
8.3.1	Overview of the Two-Layer Generation Model . . . . .	195
8.3.2	Results . . . . .	196
8.4	2.5D Sketch from Direct Estimation . . . . .	198
8.4.1	Dataset . . . . .	198
8.4.2	Model . . . . .	198
8.4.3	Results . . . . .	201

<b>9</b>	<b>Learning by Information Projection</b>	<b>203</b>
9.1	Information Projection	203
9.1.1	Orthogonality and Duality	204
9.1.2	Maximum Likelihood Implementation	205
9.1.3	The Minimax Learning Framework	206
9.1.4	Model Pursuit Strategies	207
9.2	A Unifying View	208
9.2.1	Relation to Discriminative Learning	208
9.2.2	Learning FRAME	208
9.2.3	Learning Shape Patterns	210
<b>10</b>	<b>Information Scaling and Regimes of Models</b>	<b>213</b>
10.1	Image Scaling	213
10.2	Perceptual Entropy	221
10.3	A Continuous Spectrum	224
10.4	Two Coding Schemes	224
10.5	Perceptual Scale Space	226
10.6	Perceptibility, Metastability, and the Energy Landscape	227
<b>11</b>	<b>Deep Image Models</b>	<b>235</b>
11.1	Deep FRAME	235
11.1.1	From FRAME to Deep FRAME	237
11.1.2	ConvNet Filters	238
11.1.3	FRAME with ConvNet Filters	240
11.1.4	Learning and Sampling	241
11.1.5	Learning a New Layer of Filters	242
11.1.6	Deep Convolutional Energy-Based Model	244
11.1.7	Hopfield Auto-encoder	246
11.1.8	Multigrid Sampling and Modeling	249
11.1.9	Diffusion Recovery Likelihood	251
11.1.10	Adversarial Interpretation	253
11.2	Generator Network	254
11.2.1	Factor Analysis	255
11.2.2	Non-linear Factor Analysis	256
11.2.3	Learning by Alternating Back-Propagation	257
11.2.4	EM, Density Mapping, and Density Shifting	260
11.2.5	Extracting Appearance and Geometry, Nonlinear Generalization of the AAM Model	261
11.2.6	Sparse Generator Model	265
11.2.7	Dynamic Generator model	273
<b>12</b>	<b>A Tale of Three Families: Discriminative, Generative and Descriptive Models</b>	<b>277</b>
12.1	Introduction	277
12.1.1	Three Families of Probabilistic Models	277
12.1.2	Supervised, Unsupervised, and Semi-supervised Learning	278
12.1.3	Short-run MCMC for Synthesis and Inference	278
12.1.4	Deep Neural Networks as Function Approximators	279
12.1.5	Learned Computation	279

12.1.6	Amortized Computation for Synthesis and Inference Sampling . . . . .	280
12.1.7	Distributed and Symbolic Representations . . . . .	280
12.1.8	Perturbations of KL-divergence: a Unifying Theoretical Framework . . . . .	280
12.1.9	Kullback-Leibler in Two Directions . . . . .	281
12.2	Descriptive Model . . . . .	281
12.2.1	Model and Origin . . . . .	281
12.2.2	Gradient-based Sampling . . . . .	282
12.2.3	Expressive Power . . . . .	283
12.2.4	Maximum Likelihood Estimation (MLE) and Analysis by Synthesis . . . . .	283
12.2.5	Objective Function and Estimating Equation . . . . .	284
12.2.6	Perturbation of KL-divergence . . . . .	284
12.2.7	Self-adversarial Interpretation . . . . .	285
12.2.8	Short-run MCMC for Synthesis . . . . .	285
12.2.9	Objective Function and Estimating Equation . . . . .	286
12.3	Equivalence between Discriminative and Descriptive Models . . . . .	288
12.3.1	Discriminative Model . . . . .	288
12.3.2	Descriptive Model as Exponential Tilting of a Reference Distribution . . . . .	289
12.3.3	Discriminative Model via Bayes Rule . . . . .	289
12.3.4	Noise Contrastive Estimation and Introspective Learning . . . . .	290
12.4	Generative Model . . . . .	290
12.4.1	Model and Origin . . . . .	290
12.4.2	Generative Model with Multi-layer Latent Variables . . . . .	291
12.4.3	MLE Learning and Posterior Inference . . . . .	291
12.4.4	Posterior Sampling . . . . .	292
12.4.5	Perturbation of KL-divergence . . . . .	292
12.4.6	EM Algorithm . . . . .	293
12.4.7	Noise-initialized Short-run MCMC as an Approximate Inference Engine . . . . .	293
12.4.8	Objective function and estimating equation . . . . .	294
12.5	Descriptive Model in Latent Space of Generative Model . . . . .	295
12.5.1	Top-down and Bottom-up . . . . .	295
12.5.2	Descriptive Model in Latent Space . . . . .	296
12.5.3	Model . . . . .	296
12.5.4	Maximum Likelihood . . . . .	297
12.5.5	Short-run MCMC for Synthesis and Inference . . . . .	298
12.5.6	Divergence Perturbation . . . . .	299
12.6	Variational and Adversarial Learning . . . . .	300
12.6.1	From Short-run MCMC to Learned Sampling Computations . . . . .	300
12.6.2	VAE: learned computation for inference sampling . . . . .	300
12.6.3	GAN: joint learning of generator and discriminator . . . . .	302
12.6.4	Joint Learning of Descriptive and Generative Models . . . . .	302
12.6.5	Divergence Triangle: Integrating VAE and ACD . . . . .	304
12.7	Cooperative Learning via MCMC teaching . . . . .	305
12.7.1	Joint Training of Descriptive and Generative Models . . . . .	305
12.7.2	Conditional Learning via Fast Thinking Initializer and Slow Thinking Solver . . . . .	307



# About the Authors

Zhu and Wu studied at the University of Science and Technology of China (USTC), Hefei in the late 1980s, majoring in computer science. They first met in April of 1992 at the front gate of the US consulate in Shanghai, while applying for their F1 student visas to attend Harvard University. At the Harvard Graduate School of Arts and Sciences, Zhu studied under Dr. David Mumford in Computer Science and Wu studied under Dr. Donald Rubin in Statistics, and they became roommates and collaborators. Their first joint project was the FRAME model for texture and the minimax entropy learning theory, published in 1997. After graduation, Zhu and Wu continued to carry out research on textures, textons, and sparse coding models. They reunited in 2002 as tenured faculty at the University of California, Los Angeles (UCLA), continuing work on the primal sketch model, information scaling phenomena, and on connecting these classical models and others to deep neural network structures. They received the Marr prize honorary nomination twice, in ICCV1999 and ICCV2007, for their modeling work.



Prof. Song-Chun Zhu



Prof. Ying Nian Wu

The authors believe that computer vision can be studied as a science. Inspired by the tireless pursuit of physicists over the past 300 years for a general unified theory of the physical universe, across all space and time scales, the authors believe a unified framework for computer vision is certainly possible, and it should be able to understand any potential pattern or object captured in sensory signals.

Zhu started his search for a unified theory of intelligence while in college, after encountering the book “Vision: A Computational Investigation into the Human Representation and Processing of Visual Information,” written by MIT computational neuroscientist David Marr. At the time in the late 1980s, Zhu was a student helper at a cognitive science lab at USTC. The book is considered the first attempt to outline a computational paradigm for vision, and it inspired Zhu to study computer vision with Dr. David Mumford, who also had interest in pursuing a mathematical framework for vision and intelligence. The 1990s was characterized by a transition from models based on logic and symbolic reasoning to models based on statistical modeling and computing. During this time and since then, Wu has brought valuable expertise in statistics to the team. By the late 1990s, Zhu and Wu had adopted a Bayesian view and introduced statistical methods to the computer vision community.

Notably, the pursuit of a unified framework for computer vision dates back to Ulf Grenander, a pioneer who launched General Pattern Theory in the 1960-70s while at the Division of Applied Math at Brown University. Zhu studied pattern theory as well, with a group as a postdoc, and benefitted greatly from

interactions with this group. On this note, an underlying goal of this book is to apply the methodology of general pattern theory to formulating concepts in Marr's paradigm.

At the Center for Vision, Cognition, Learning, and Autonomy at UCLA, Zhu and Wu have supervised many Ph.D. students and postdocs who have both contributed their thesis work as examples in this book and assisted with editing.



# Preface

*“The highest activity a human being can attain is learning for understanding, because to understand is to be free.”*

– Baruch Spinoza, *Philosopher*, 1632-1677.

## Story of David Marr

This textbook is intended for researchers and graduate students in statistics, computer science, and engineering. Based on contributions from multiple authors in the past 30 years in the Department of Statistics and the Department of Computer Science at the University of California, Los Angeles, it may be used as a reference in the fields of computer vision and pattern recognition, machine learning, graphics, robotics, and artificial intelligence. As the first book of a three-part series, this book is offered as a tribute to pioneers in vision, such as Béla Julesz, David Marr, King-Sun Fu, Ulf Grenander, and David Mumford. In this book, the authors hope to provide foundation and, perhaps more importantly, further inspiration for continued research in vision.

An overarching goal of the three-book trilogy is to provide unifying mathematical foundations for research within vision, cognition, language, and autonomy in artificial intelligence. With the rise of deep learning, applications of neural networks continue to grow but still very much at the expense of understanding how these models truly work or generate their solutions. It is all-too-common to refer to neural networks as “black boxes,” representing a common lack of understanding in the community. In an effort to guide wise uses of neural networks, and to promote the unification of various artificial intelligence theories under a common framework, the vision models in this book take inspiration from many authors in artificial intelligence and other fields, such as statistics, physics, neuroscience, and psychology. Models in this book may establish more carefully designed uses of neural networks, or in some cases may help to modify or generalize existing neural networks, so that more explicit and more efficient models, in terms of both data efficiency and computational efficiency, may emerge.

David Marr is well-known for pioneering a resurgence of interest in computational neuroscience and for integrating the fields of psychology, neurophysiology, and artificial intelligence in his research on visual processing. After he passed away at the age of 35, his work was published posthumously in 1982 in the book *Vision: A computational investigation into the human representation and processing of visual information* [153]. Although Marr published early works on the cerebellum in 1969, neocortex in 1970, and hippocampus in 1971, he is most well-known for his research in vision. He laid foundations for continued studies in vision for various fields, such as computational neuroscience and computer vision.

Referred to as Marr’s tri-level hypothesis, Marr viewed vision as an information-processing system that

should be understood at three distinct, but complementary, levels: the computational level, the algorithmic or representational level, and the implementational or physical level. At the computational or mathematical level, one seeks to understand the problems the visual system solves and, in a similar sense, why it solves them. At the algorithmic or representational level, one seeks to understand how the visual system solves its problems, i.e., the representations it uses and the algorithmic processes it employs to manipulate those representations. At the implementational or physical level, one seeks to understand how the visual system is physically realized. In the same spirit as Marr's tri-level hypothesis, in this book vision problems are addressed with respect to each of the computational, representational, and implementational levels. Crucially, vision problems may be studied at the computational and representational levels, independent of the implementational level, which is often realized through neural networks [154].

Marr believed that a deep understanding of the brain entails an understanding of the problems it encounters, i.e., the input, and how it solves them, i.e., the steps taken to produce the output. As his interests gradually evolved from the brain to visual processing, he began to treat vision similarly. He described vision as a computational process that takes as input a two-dimensional array on the retina and outputs a three-dimensional description of the world. His three stages of vision, depicted in Fig. 1, include a primal sketch, a 2.5D sketch, and a 3D model. A 2.1D sketch was also proposed by Nitzberg and Mumford [169].

A primal sketch extracts key components of a scene, such as simple edges and regions. Textures and textons are constituent parts of a primal sketch, which correspond to early stages of human visual perception, i.e., the first visual phenomena noticed by humans when viewing an object. A 2.5D sketch reflects textures and depth. The 2.1D and 2.5D sketches correspond to mid-level vision. In a complete 3D model, a scene is visualized in a continuous 3D map. The 3D model corresponds to high-level vision and provides an object-centered perspective, while the primal, 2.1D, and 2.5D sketches provide human, view-centered perspectives.

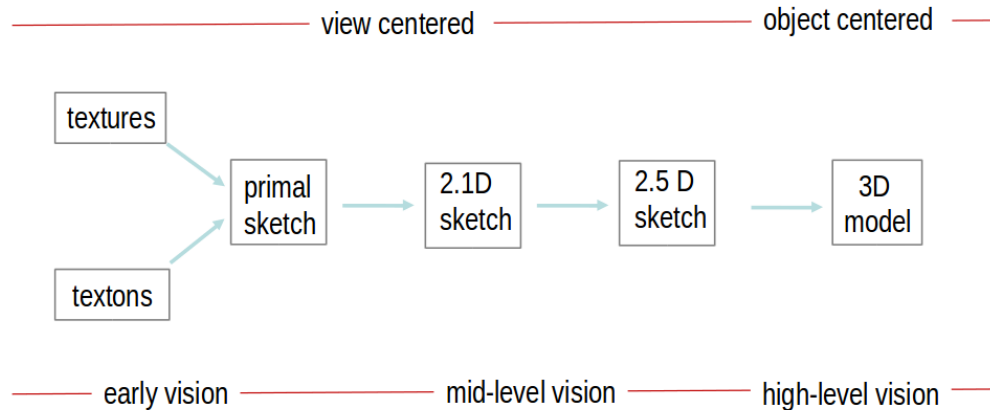


Figure 1: Stages of vision according to Marr's paradigm

Various research influenced Marr's book *Vision*, such as psychology experiments by Béla Julesz on textures and textons, neuroscience discoveries in edge detection and filters, the random dot stereopsis by Julesz, the shape-from-shading theory from Berthold K. P. Horn, and research on generalized cylinders and 3D representations. Various works facilitated the transitions from primal sketch to 2D sketch to 2.5D sketch to a complete 3D model. For a primal sketch formulation, work on textures and textons paired with work on edge detection and filters provided crucial foundations. For the 2.5D sketch, the random dot stereopsis from Julesz and the shape-from-shading theory by K. P. Horn facilitated the modeling of textures and depth. Lastly, for the 3D model, work on generalized cylinders and 3D representations made it possible to model scenes in continuous, 3D maps.

As a whole, Marr’s work proposed a holistic framework for understanding vision and touched on broader questions about how cognition may be studied. Thirty years later, the main problems that occupied Marr remain fundamental, open problems in the study of vision. This book may be understood as a bridge between Marr’s theory of vision and the modern treatment of computer vision with statistical models. It explores deep connections between Marr’s ideology and neuroscience discoveries, while solidifying such discoveries with mathematical models.

The unifying mathematical foundations expounded in this book also counteract a “big data for small tasks” paradigm that dominates the deep learning community today. This paradigm refers to the practice of exploiting massive amounts of data for highly specific tasks and essentially repeating this process for each new task. A great deal of research readily falls susceptible to such a “task trap,” such that solutions often do not contribute to a greater, unified framework for vision. The methods in this book are presented under a unified framework, with the hope that it helps advance progress toward a more integrated vision community.

Given the wide variety of artificial intelligence methods used today on similar problems, it is clear much effort is devoted by the community as a whole to often entirely different research approaches. Undoubtedly, ingenuity and novelty contribute greatly to research advancement, but the community as a whole could work more in harmony and hence more efficiently if it adopted a common ground, or overarching framework, for general-purpose research in vision. This type of maturation for the field of vision is comparable to a process other academic fields have undergone and is arguably long overdue.

## Beyond David Marr’s Paradigm

This book provides mathematical foundations or tools for many of Marr’s concepts. Notably not available during Marr’s time, they include Markov random fields; the FRAME (Filters, Random fields, and Maximum Entropy) model [263], i.e., a predecessor of the energy-based model (EBM); generative models; sparse coding models [174]; various inference algorithms; and deep neural networks. Textures and textons are fundamental elements of Marr’s paradigm, in particular for the primal sketch. For textures, Markov random fields and the FRAME model, which utilizes Markov random fields, may be used to mathematically formulate a Julesz ensemble. For textons, generative models and sparse coding models provide essential mathematical formulations.

Béla Julesz was a psychologist at Bell Labs and a professor at Rutgers best known for his work on random dot stereo and texton theory. Although both theories have their weaknesses, his work inspired many thinkers, including Marr, to establish the field of computational vision. Providing a mathematical definition for a Julesz ensemble for textures, the FRAME model offers the following conclusion: the limit of the FRAME model, as an image domain grows to infinity, is a Julesz ensemble. Let us begin to understand this.

A Julesz ensemble  $\Omega(h)$  is the limit of an equivalence class  $\Omega_D(H)$  as  $D \rightarrow Z^2$  under some boundary conditions, in which  $D$  is the image domain and  $Z^2$  is the entire image space. An equivalence class  $\Omega_D(H)$  is defined as  $\Omega_D(H) = \{I : h(I) \in H\}$ , in which  $I$  is the image and  $h = (h^{(i)} : i = 1, 2, \dots, k)$  is a set of normalized statistics for images on a finite lattice. As the image lattice  $D$  for the equivalence class  $\Omega_D(H)$  of images grows to infinity  $Z^2$  in a 2D plane, statistical fluctuations diminish and the image set becomes deterministic. Thus, the Julesz ensemble  $\Omega(h)$ , which is the limit of the equivalence class as the image lattice grows to infinity, may be defined by a simple uniform distribution  $q(I; h) = 1/|\Omega(h)|$  for  $I \in \Omega(h)$  and  $q(I; h) = 0$  otherwise. Hence, the Julesz ensemble provides a definition of texture on a 2D plane that is associated with a uniform distribution  $q(I; h)$ .

The FRAME model provides a definition of texture on a finite lattice that is associated with a conditional distribution  $p(I; \beta)$ , in which  $\beta$  are 1D potential functions for the selected filters of the model. The distributions of the FRAME model and the Julesz ensemble are indeed consistent with each other, reflective

of the fact that both models respect the minimax entropy principle. The minimax entropy principle both minimizes and maximizes entropy in a two-step learning process that (1) chooses informative features, or statistics, to minimize entropy, i.e., the log-volume or uncertainty of the system, and (2) under these constraints, chooses a distribution that maximizes entropy, i.e., minimizes bias. To reiterate the conclusion of the FRAME model now with more vocabulary, the limit of a FRAME texture model  $p(I_D|I_{\partial D}; \beta)$ , in which  $I_{\partial D}$  is the neighborhood of pixels around the pixel of interest as an image domain  $D$  grows to infinity  $Z^2$ , is a Julesz texture ensemble  $f(I; h_c)$ , in which  $c$  represents a concept.

To summarize, the equivalence of the Julesz ensemble and the FRAME model follows from the following theorems: (1) for a very large image from the Julesz ensemble,  $I \sim f(I; h_c)$ , any local patch of the image  $I_D$ , given its neighborhood  $I_{\partial D}$ , follows a conditional distribution specified by a FRAME model  $p(I_D|I_{\partial D}; \beta)$ , and (2) as the image lattice  $D$  grows to infinity  $Z^2$ ,  $f(I; h_c)$  is the limit of the FRAME model  $p(I_D|I_{\partial D}; \beta)$  in the absence of a phase transition.

So although the FRAME model is defined on a finite lattice  $D$ , as this lattice  $D$  grows to infinity  $Z^2$ , it may be shown mathematically that the FRAME model generalizes to the Julesz ensemble. This demonstrates an important relationship between conceptualization and modeling; concepts by nature are defined on an infinite domain but in practice can only be modeled on a finite domain. That is, concepts are inherently defined by the collection of all their instances in the universe, but realistically they can only be modeled, or represented, using a finite set of these instances. It is impossible to have access to all the data in the universe for a particular concept. A texture defined on an infinite lattice  $Z^2$ , modeled by  $h_c$  according to a Julesz ensemble  $f(I; h_c)$  defined on a finite lattice  $D$ , could ultimately be modeled by, e.g., the 1D potential functions  $\beta$  and filters of the FRAME model  $p(I_D|I_{\partial D}; \beta)$ .

For modeling textons, sparse coding models and active appearance models (AAM) have found great success. Sparse coding states that objects may be represented by the strong activation of a relatively small set of nodes, or neurons. It is motivated in part by research in biological vision. For example, Huber and Wiessel in the 1960s performed experiments on cats to record the activations of cortical cells in the V1 section of the mammalian brain. It was observed that bars of light oriented mostly vertical activated a particular set of neurons in the V1 section of the brain, while the same bar of light oriented in other directions failed to activate the same neurons. In another neurophysiological experiment, a neuron was recorded to selectively fire only to images of Jennifer Aniston, inspiring a sparse coding scheme called grandmother cell coding. Furthermore, some neurons referred to as mirror neurons fire not only when performing an action, but also when observing other subjects perform the same action. In general, one of the striking observations about physiological recordings from sensory cortical areas of the brain is the difficulty of finding stimuli that effectively activates some given neurons. These difficulties reflect the narrow functionality of cortical neurons which, given the incredibly large number of them, suggests that a sparse coding system may be in place in the brain.

Field [51] performed experiments on the primary visual cortex to suggest that basis functions limited in both space and frequency domains, such as Gabor functions, maximize sparseness when applied to natural images. Olhausen and Field [171] gave examples of sparse coding in other brain regions. In [174], Olhausen and Field defined an explicit objective function that promoted both high sparseness and low reconstruction error. The minimization of this function on natural images leads to a set of basis functions that resemble localized receptive fields of simple cells in the primary visual cortex. Sparse coding is closely related to independent component analysis [13].

Sparse coding is also closely related to active appearance models (AAM), another mathematical tool for defining textons. An AAM is a vision algorithm for matching a statistical model of object shape and appearance to a new image. It is related to the active shape model (ASM), for which a disadvantage is that it only uses shape constraints, together with some information about the image structure near the landmarks.

The ASM does not take advantage of all available information, e.g., texture across the target object, which can be modeled using an AAM.

Various inference algorithms and deep neural networks, also methods not available during Marr's time, are widely used throughout this book. Of course, the vision community has seen incredible success in recent years due to advances in deep learning, but as mentioned before, research efforts in vision are often devoted to entirely different approaches. In this sense, one could argue that the soul or direction of vision has been lost to some degree, and the field could gain a great deal by adopting more common mathematical grounds. In an effort to recapture the soul of vision, this book carefully establishes mathematical foundations for general-purpose research in vision by examining all theories in the literature, beginning with Marr's paradigm and progressing up to the most recent uses of neural networks.

## **Introducing the Book Series**

The book series consists of three parts.

The first book, introduced here, covers David Marr's paradigm and various underlying statistical models for vision. The mathematical foundations herein integrate three regimes of models (low-, mid-, and high-entropy regimes) and provide essential foundation for research in visual coding, recognition, cognition, and reasoning. Concepts in this book are first explained for understanding and then supported by findings in psychology and neuroscience, after which they are established by statistical models and further linked to research in other fields such as physics. A reader of this book will gain a unified, cross-disciplinary view of artificial intelligence research in vision and will accrue knowledge spanning from psychology to neuroscience to statistics.

The second book defines a stochastic grammar for parsing objects, scenes, and events, posing computer vision as a joint parsing problem. It summarizes research efforts over the past 20 years that have worked to extend King-Sun Fu's paradigm of syntactic pattern recognition. Similar to David Marr, King-Sun Fu was a pioneer and influential figure in the vision and pattern recognition community.

The third book discusses visual commonsense reasoning, including subjects such as functionality, physics, intentionality, causality, and values. The third book connects vision to cognition and artificial intelligence.

The authors would like to thank many current and former Ph.D. students at UCLA for their contributions to this book: Erik Nijkamp, Eric Fischer, Jonathan Mitchell, Linqi Zhou, Mitchell Hill, Yaxuan Zhu, Ruiqi Gao, Yuxin Qiu, Chi Zhang, and Tianfu Wu. Erik Nijkamp and Eric Fischer, especially, have worked extensively on polishing the manuscript and writing Chapters 1, 10 and 11.



# 1

## Introduction

A primary aim of this book is to pursue unified knowledge representation for vision. In a world in which technology generates immense amounts of data from a wide spectrum of sources, it is of growing importance to establish a common framework for knowledge representation, learning, and discovery. A current problem in artificial intelligence is how to acquire truly massive amounts of knowledge, akin to the faculty of common sense in humans, from raw sensory signals and, moreover, use this knowledge for inference and reasoning. In this book, the vision models presented for acquiring such knowledge are based most fundamentally on statistical properties discovered for natural images over the past several decades. Similar to models in physics, models are advanced based on empirical grounds, such as discovering additional patterns in data. In this way, models stay free of bias.

Discriminative modeling approaches are not discussed in this book in-depth. In statistical classification, two main approaches exist: discriminative and generative modeling. In discriminative modeling, the conditional probability of a target  $Y$  is modeled given an observation  $x$ , i.e.,  $P(Y|X = x)$ . Often, the value of a target variable  $Y$  is determined by training a model on thousands of examples. A discriminative classifier may perform well at discriminating between, e.g., chairs and non-chairs, but does it truly understand the concept of a chair? Consider adversarial attacks in the literature, in which an image that a discriminative classifier can otherwise correctly classify with a high degree of certainty is injected with a small amount of noise, in many cases barely perceptible to a human, and the predicted class is entirely different, e.g., a bus instead of a chair. This behavior demonstrates that a discriminative model does not actually learn the concept of a chair as a human does – it only learns to discriminate between chairs and non-chairs using whatever obscure image features it may prefer.

In cases in which discrimination is not the ultimate goal, generative modeling approaches, explored in this book, facilitate the construction of models more representative of human learning. In generative modeling, given an observable variable  $X$  and a target variable  $Y$ , the joint probability distribution  $P(X, Y)$  is modeled. From this, the conditional probability  $P(Y|X = x)$ , modeled directly in discriminative modeling, is computed. Generative modeling is more indirect but also more probabilistic in comparison to discriminative modeling, allowing more domain knowledge and probability theory to be applied. In this way, generative models better represent human learning. For example, with prior domain knowledge encoded, a generative model can generalize knowledge to new but related tasks. And this is akin to human learning; a human may be able to intuit, for example, how to make orange juice with the prior knowledge of how to make lemonade.

After discussing statistical properties discovered for natural images in Chapter 2, the most basic units of visual perception, textures and textons, are introduced in Chapters 3 and 4, respectively. A texture could be sand, or the thousands of leaves of a tree viewed from afar, as in Fig. 1.1, for which individual components in



Figure 1.1: One can observe, as an example of texture, the tree leaves in the background and, as an example of textons, the edges created by tree trunks in the foreground.

preattentive vision are not disentangled. Preattentive vision is characterized as human vision before focusing on any specific region of some visual stimuli. A texton, or a “token” as referred to by Marr, can be thought of as a most basic element, like a bar, edge, corner of an eye, or trunk of a tree as in Fig. 1.1. Textons form the structural part of an image and object boundaries. Note that the same object may be perceived as textons or texture, depending on potential viewing distance or focal point, and in natural scenes these two entities are seamlessly interweaved. Following David Marr’s insight, these two modeling components may be integrated to form a generative image representation called primal sketch, which he referred to as a “symbolic” image representation in terms of image primitives. Primal sketch is discussed further in Chapter 6, after a discussion of Gestalt laws and perceptual organization, which naturally follows from Chapter 4 on textons.

Marr defined different stages of vision, starting from a 2D visual array on the retina to a 3D description of the world. His stages of representation include a primal sketch, a 2.1D sketch, a 2.5D sketch, and a 3D model. The first two stages perform separation between four main factors known to influence perceived visual intensity: geometry (shape and position), reflectance of visible surfaces, illumination, and viewpoint. A primal sketch extracts fundamental components of the scene, like edges and regions, and it looks like a pencil sketch. In a primal sketch, geometric structures, pixel intensity changes, and illumination effects are detected. A primal sketch captures spatial layout by using textons, such as edges, bars, and blobs. A 2.1D sketch introduces layered representations of the input image and is discussed in Chapter 7. A 2.5D sketch, discussed in Chapter 8, represents orientation, depth, and textures, considering distance from the viewer and discontinuities in depth and surface orientation. Both primal and 2.5D sketches are viewer-centered perceptions. In a 3D model, the scene is visualized in a continuous 3D map, as an object-centered perception. The 3D representation describes shapes and their organization using a hierarchical organization of volumetric and surface primitives. Marr’s stages of visual perception serves as a basis for further analyses in the book.

In Chapter 9, information projection is introduced as a generic framework for learning a statistical model as an approximation to the true data distribution. In Chapter 10, information scaling and regimes of models are discussed, including entropy, metastability, and energy landscapes. In Chapter 11, image models with multilayer neural networks, such as Deep FRAME, the Hopfield auto-encoder, and short-run



MCMC (Markov Chain Monte Carlo) are presented. In Chapter 12, three main families of machine learning models, i.e., discriminative, generative, and descriptive models, are further examined.

## 1.1 Goal of Vision

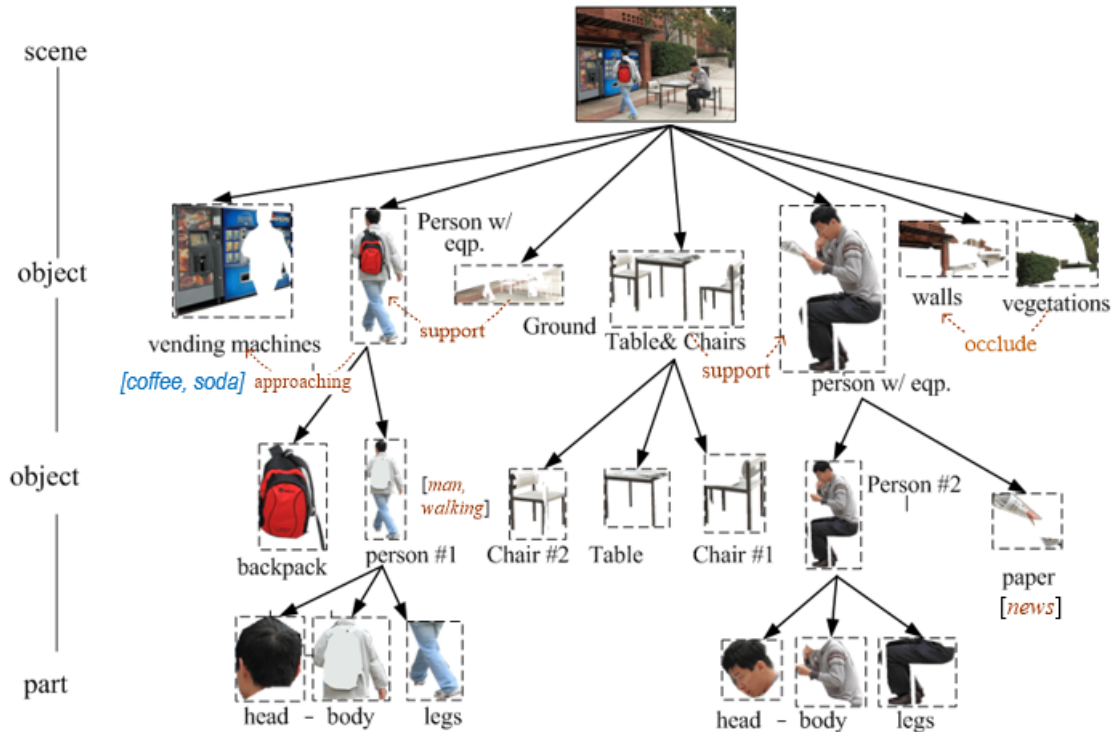


Figure 1.2: Scenes and events may be understood in terms of a parse graph. From a single image, a dense 3D scene may be reconstructed, by estimating camera parameters, materials, and illumination. The scene may be parsed hierarchically in terms of relations, intents, beliefs, attributes, and fluents. The actions of agents may be predicted over time and hidden object states, e.g., water boiling inside a metal kettle, which humans can naturally infer from the image, may even be recovered.

In the parse graph in Fig. 1.2, there can be several types of nodes including scenes and objects, minds and intents, hidden objects, actions, imagined actions, attributes, and fluents, i.e., how objects change over time. Nodes are organized in a tree-like hierarchical structure with potential connections between sibling nodes. Parse graphs for knowledge representation in vision will be discussed in greater detail in the second book, but for now it may be understood that it involves parsing scenes and events in a picture, or a video sequence, into nodes such as in Fig. 1.2.

These nodes may be represented with words one might use to describe them in natural language, e.g., “backpack” or “vending machines,” or they may be represented with “words” that are not really words at all – perhaps a symbol, number, some other character, an expression in sign language, or a facial expression. Either way, for simplicity, the descriptors of these nodes are referred to as words. They can be thought of as *visual* words in the sense that even a primate, for example, could have a word for a node, even though it is not endowed with a capacity for language. Sometimes in this book, words are referred to as symbols or



and objects. The dorsal pathway, or “vision-for-action” pathway, leads to the parietal lobe, which is involved with processing objects’ spatial locations. This pathway has been tied to movements such as reaching and grasping, which are based on evolving spatial locations, shapes, and orientations of objects. Understanding the difference between the two visual pathways, the ventral pathway for perception and the dorsal pathway for action, is useful knowledge for vision research and for understanding the inspiration behind theories presented later on.

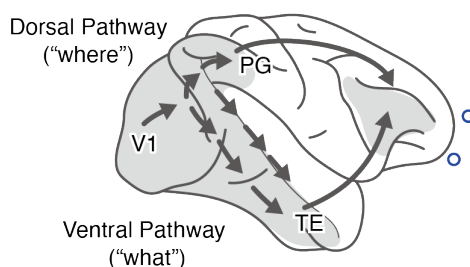


Figure 1.4: The dorsal pathway is responsible for the spatial aspect of vision, i.e., “where” humans see objects. The ventral pathway is responsible for the main content of vision, i.e., “what” objects humans see.

Another consideration for vision research that must be accounted for is the fact that human vision is often task-driven. After viewing an instructional video about how to make watermelon juice, one could most likely reenact the process, including chopping watermelon, blending it, and so forth. The physics and the functionality of the objects, and potentially hundreds of other subtasks involved in making the juice, may be easily understood. Humans additionally understand the causality and how to, e.g., switch the order of the steps without affecting the final result. From a small amount of data, i.e., the instructional video in this case, humans can absorb a massive amount of information, forming what is actually just referred to as common sense. Humans possess this remarkable ability because vision is incredibly task-driven, absorbing specific information from complex visual stimuli.

Thus, the faculty of common sense, which forms part of the prior knowledge each human possesses, ultimately entails an understanding of physics, functionality, causality, intentionality, and utility, amongst other properties of a scene. Pursuing holistic knowledge and learning reminiscent of prior knowledge and learning in humans may be referred to as the “small data for big tasks” paradigm. This is where efforts are concentrated in this book. Many avenues in deep learning today, unlike human learning, depend on massive amounts of data to learn highly specific tasks, such as the task of making watermelon juice or identifying a person from his or her face in facial recognition. This assumes no prior knowledge and this type of learning is not generalizable. As mentioned previously, this contrasting and all-to-common paradigm in deep learning may be referred to as the “big data for small tasks” paradigm.

As a given task or purpose very much influences the interpretation of visual stimuli to form knowledge, similar behavior should be exhibited by vision models. Accordingly, in this book the optimal model and knowledge representation of some visual data depends not only on the data but also on the relevant task. Here, a contrast may be made with the field of physics. In vision, there are subjective, in addition to objective, considerations that guide the formation of models. As a vision model seeks structure in the data, i.e., an image space, the optimal structure or architecture of the model depends on its given purpose. As an example, the hierarchical structure of an And-Or-Graph depends on its purpose, such that it evolves as the purpose of the model evolves.

As mentioned, vision models should also generalize well. Although excellent for a specific task, discriminative modeling approaches clearly lack generalization, which does not reflect human learning. Again,

a human could likely infer how to make orange juice after learning how to make lemonade or watermelon juice. Or a human could infer how to crush a walnut with a book after observing someone crush it with a hammer. Humans have the ability to generalize knowledge to novel but similar tasks, often after learning from just one example. The generative modeling approach offers the tools to begin to mimic this incredible learning ability.

Humans generalize knowledge well primarily due to prior knowledge and imagination, amongst other faculties. From birth, humans continually gain a better understanding of space, time, causality, functionality, and other features of the world, utilizing that knowledge each time a new task is learned. As humans learn, they add to their prior knowledge, in this way creating a continual accumulation of it. With the imagination, humans have the ability to extrapolate prior knowledge to imagined future or hypothetical tasks, a powerful tool for generalizing knowledge.

## 1.2 Seeing as Bayesian Inference: Vision is Ill-Posed and Mostly an Illusion

Clearly, the way in which humans form and represent knowledge from visual stimuli is complex and relies on more than just vision—it is dependent on prior knowledge, imagination, and in general the mind, or agent, that perceives. In fact, every image has literally infinite interpretations, and humans must derive only one or two meaningful interpretations for each one. Vision, in this sense, is under-constrained and necessitates some form of guessing by the agent, using prior knowledge. Images, by themselves, lend minimal information. This is an important point and essentially what it means for vision to be ill-posed and mostly an illusion. Interpretation of visual stimuli is largely independent of the visual stimuli itself—it depends on the agent’s prior knowledge and imagination.

It has been shown that a form of top-down processing performed by the visual system also contributes greatly to visual interpretation. Top-down processing makes use of global information propagation, exploiting high-level knowledge possessed by the agent. The agent considers high-level contextual information and performs top-down inferential reasoning based on prior knowledge, both innately acquired through evolution and learned through experience. With top-down processing, the agent may form an interpretation of a visual stimuli, even if the stimuli was only partially perceived or perceived poorly, e.g., due to poor lighting conditions or quick movement. To give an example in vision, top-down processing may help one intuit the object or scene of a partially completed puzzle, although only portions of what typically represent the object or scene are present. To give an example in language, top-down processing eases the understanding of sloppy handwriting given complete sentences, as opposed to only isolated words, as surrounding words provide context to aid understanding. With the role of top-down processing in the visual system, top-down architectures have become paramount to the design of learning and inference algorithms in vision.

Given prior knowledge, a basic assumption since the time of Helmholtz’s research in the 1860s is that a given visual input may be represented as the most probable computed interpretation of an image. To give a Bayesian formulation for vision, let  $I$  be an image and  $pg$  be a semantic representation of the world, such as a parse graph. The most probable representation  $pg^*$  of the image is defined as  $pg^* = \arg \max_{pg \in \Omega} p(pg|I) = \arg \max_{pg \in \Omega} p(I|pg)p(pg)$ . To obtain the most probable representation  $pg^*$ , it is necessary to sample from the posterior  $p(pg|I)$  and obtain candidates  $(pg_1, pg_2, \dots, pg_k) \sim p(pg|I)$ . This is a crucial point—sampling from the posterior probability lends the possible visual interpretations. The result of this process,  $pg^*$ , represents the most probable semantic representation of the image and the visual interpretation of the agent.

The quality of the prior knowledge of a model may be judged using a method referred to as analysis by synthesis, based on synthesized examples. Analysis by synthesis can be compared to a Turing test as a way to judge the humanlike thinking capability, or intelligent behavior, of an artificial intelligence model. In a Turing test, a subject asks questions to two different agents, one a machine and the other a human,

and the machine is determined to be more humanlike the more uncertain the subject is in identifying the human from the machine, based on each of their responses. Thus, the machine does not need to have correct responses; it merely needs to respond like a human, to perhaps fool the subject into thinking it is, in fact, the human. The analysis by synthesis method may also be used to judge humanlike thinking capability, but in a way that does not depend on input, such as a subject asking questions. This eliminates the possibility of any questionable input. The analysis by synthesis method thus provides an unrestricted way to judge what a machine knows, i.e., its prior knowledge—in a way, this is similar to dreaming in humans. Dreaming, or the imagination, is also unrestricted in the sense that these are perceptions, often of great visual detail, that also receive no input visual stimuli. Interestingly, then, the analysis by synthesis method may be viewed as a way to determine what a machine “imagines” or “dreams.” If one were to draw several random samples meant to represent a dream from a completely untrained model that possesses no prior knowledge of the world, the random samples would be akin to white noise. A well-informed model, however, would produce a detailed, interesting dream.

The fact that vision relies so heavily on prior knowledge and the imagination signifies that it is highly probabilistic. Accordingly, probability and Bayesian inference play large roles in vision research. The probabilistic nature of vision is most obvious with problems exhibiting imperceptibility, in which human vision jumps between image interpretations often entirely different. In the bikini versus martini example, a well-known example demonstrating ambiguity in visual perception, human vision oscillates between interpretations of a bikini bathing suit and a martini drinking glass, depending on how the visual system interprets the image. Mathematically, the fact that perception may jump between interpretations in some structured state space poses a significant challenge for the design of learning and inference algorithms. Ambiguities in visual inference that occur with human vision are, in general, difficult to model. Imperceptibility in vision is more formally introduced in Chapter 9, which discusses information scaling regimes.

Imperceptibility may be formally defined as the complexity of an image subtracted from the complexity of the world. As a measure of complexity, it may be aptly modeled with entropy. Given a generative model such that  $p_g \sim p(p_g)$  and  $I = g(p_g)$ , in which  $p_g$  is a semantic representation of the world and  $I$  is an image, imperceptibility  $H(p(p_g|I)) = H(p(p_g)) - H(p(I))$ . Entropy measures complexity or uncertainty, e.g., the image complexity  $H(p(I)) = \mathbb{E}[-\log p(I)] = -\int p(I) \log p(I) dI$ . Hence, as world complexity increases relative to image complexity, imperceptibility grows. When the posterior probability  $p(p_g|I)$  exhibits high imperceptibility, it means that certain variables in  $p_g$ , the semantic representation of the world, cannot be inferred as the uncertainty is too high; hence, they are imperceptible. In this case, the model’s representation of the world would need to be reduced in complexity.

As the concept of imperceptibility explains ambiguities in inference, it has been utilized in analyses of abstract art. Artists intentionally create imperceptible aspects of a piece of artwork to induce ambiguous visual interpretations. Viewers may proceed down different perceptual paths to interpret the artwork in various ways.

### 1.3 Knowledge Representation

To begin to understand the way knowledge may be represented in vision, first consider the space of all image patches of natural scenes and of a fixed size, e.g.,  $10 \times 10$  pixels. These image patches reside in an image space whose dimension is the total size of the image patches, in this case  $10 \times 10 = 100$ . Each image patch can be treated as one point in the image space. Hence, the image patches together form a population of points in the image space. One may consider an analogy between this population and the 3D universe, as illustrated by Fig. 1.5. The distribution of mass within the universe is highly uneven. There are high densities of mass at stars, but there are low densities of mass across nebulas (clouds of dust and gas in outer





Figure 1.5: Consider the complex distribution of mass in the universe, e.g., at stars (high density, low volume clusters) and over nebulas (low density, high volume regions). Left: the universe with galaxies, stars and nebulas. Right: a zoomed-in view of a small part of the image.

space), in which mass is spread out. The distribution of the population of natural image patches in the image space is also highly uneven. To model this space, it becomes necessary to identify, map out, and catalog high-density clusters such as stars, as well as low-density regions such as nebulas.

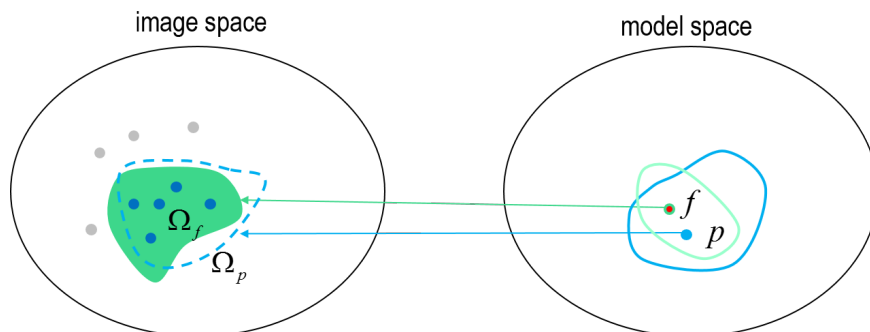


Figure 1.6: The hope is that  $\Omega_p$ , the learned estimate of  $\Omega_f$ , approximates as closely as possible  $\Omega_f$ . The better the approximation, the more closely the learned concept  $p$  approximates, and thus accurately represents, the real concept  $f$ .

Just as a representative in the United States Congress represents a subset of the population, mathematically, a concept represents a subset of image patches within the entire universe of image patches. All the image patches in this subset are perceived as the same pattern and hence are described by the same symbol, or “visual word.” The subset, or subpopulation, of image patches that correspond to a concept can be represented mathematically by a probability distribution or a statistical model. Hence, concepts can ultimately be represented by statistical models, which is quite powerful.

Thus, there are two spaces: the image space of image patches (signals) and the model space of concepts (symbols), depicted in Fig. 1.6. Each concept, which corresponds to a set of image patches in the image space, is represented as just one point in the model space. For example, in Fig. 1.6 the set of image patches  $\Omega_f$  in the image space correspond to one concept  $f$ , a point in the model space. All the image patches of a concept, despite having diverse pixel intensities, correspond to the same concept because they are perceived as the same pattern or object by the visual system.

Suppose a number of image patch examples are observed from the set of image patches  $\Omega_f$ , and the

goal is to learn a concept  $f$  in the model space based on these examples. In other words, the goal is to recover the set  $\Omega_f$  to best learn the concept  $f$ . First and foremost, it is impossible to fully recover the set  $\Omega_f$ , since it is impossible to access all image patches in the universe that belong to  $\Omega_f$ . All the image patches that correspond to the concept of sand, for example, are clearly not somewhere in a dataset. However, it is still possible to come up with an approximated estimate of  $\Omega_f$ . Call it  $\Omega_p$ , which corresponds to a concept  $p$  in the model space. The hope is that the set  $\Omega_p$  is close to the set  $\Omega_f$  in the image space, so that the concept  $p$  will be as close as possible to the concept  $f$  in the model space. For example, one might hope  $p$  is within a distance  $\epsilon$  of  $f$ , measured by some predefined metric. In machine learning, identifying  $\Omega_f$  based on examples from  $\Omega_f$  is referred to simply as learning, and these examples are training examples.

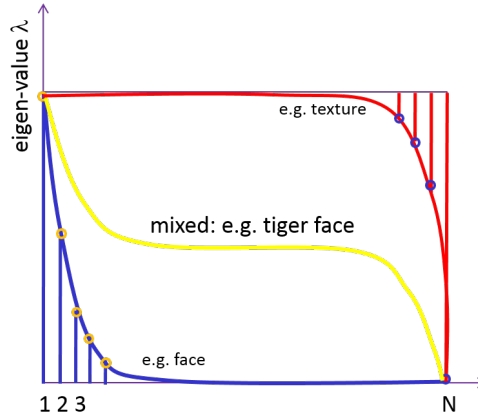


Figure 1.7: Observe a plot of eigenvalues, in decreasing order, both for concepts of low-dimension (blue curve) and high-dimension (red curve).

Concepts can differ greatly in their complexities. Some concepts may appear to be very simple or regular, such as a line segment, a triangle, or even a human face, while other concepts can appear quite complex or random, such as stochastic textures like grasses or other foliage. Some concepts lie in between, such as the face of a tiger. Ultimately, concepts may be defined as textures, textons, or some composition of the two. Compositions of textures and textons are discussed in the second book in the series. Fig. 1.7 is a simple illustration of concepts with different complexities. For each concept, such as a texture or the face of a tiger, a number of examples are collected, which may then be aligned in the case of, e.g., faces. Principal component analysis is performed and the eigenvalues are plotted in descending order. For a simple concept such as a face, the eigenvalues drop to 0 very quickly, indicating that the images lie in a very low-dimensional space. For a complex concept such as a texture, the eigenvalues stay high for an extended range, indicating that the images lie in a very high-dimensional space. For the face of a tiger, the eigenvalues lie in between.

The complexity of a concept  $f$  in the model space can be measured by the log-volume of the corresponding subset  $\Omega_f$  in the image space. This log-volume is called entropy in statistical physics and information theory, and it is called intrinsic dimension in mathematics and coding theory. As one can see in Fig. 1.8, entropy may be used as an axis on which to map concepts. Geometric patterns belong to the low-entropy regime, while texture patterns belong to the high-entropy regime. Many object patterns lie in the mid-entropy regime. Textons are low-entropy, textures are high-entropy, and many compositional structures in between are mid-entropy. Textures, textons, and compositions provide three ways to characterize a set in the image space that represents a concept in the model space. While patterns in the low-entropy regime tend to be simple and patterns in the high-entropy regime tend to be random, patterns in the mid-entropy



Figure 1.8: More simple geometric patterns like edges and bars have low entropy, while stochastic textures like fur and carpet have high entropy.

regime tend to be quite informative. The informativeness of a concept  $f$  can be measured by the number of parameters needed to specify it in the model space. Central to the goal of this book is to establish a unified mathematical framework and form of knowledge representation under which these entropy regimes may be further studied and defined.

With the aim of unified knowledge representation, a natural question is how to relate different types of knowledge. Vision models that lend knowledge representation can generally be categorized into three paradigms: logic models, probabilistic models, and discriminative models, as shown in Fig. 1.9. In the 1960s and 1970s, knowledge was mostly represented by logic formulas, i.e., propositions and predicates, called well-formed formulas. In this paradigm, a concept is represented by a set, which is in turn specified by well-formed formulas. As such, a concept is ultimately equivalent to a set of states, from a joint state space, that satisfies all the well-formed formulas. Propositional calculus, first-order predicate calculus, situation calculus, and event calculus are all used to represent knowledge. In general, the logic paradigm for knowledge representation is alive and well, and it is used today to solve many problems in vision. For example, situation calculus is used for causal reasoning in robot planning. A clear advantage of logic for knowledge representation is that it is rigorous and enables pure reasoning; however, a clear disadvantage is that it is inherently fragile in that it is not grounded on observational data. Truths found by exercising logic alone are called *a priori* truths; they are justified by reasoning that proceeds only from theoretical deduction, completely independent of empirical data.

Beginning in the 1980s, based partly on a desire to ground knowledge representation on empirical data, there was more research into probabilistic models, which provided imperfect knowledge representation and the ability to make inferences from observed signals with some degree of uncertainty. Note that “imperfect” knowledge representation does not imply that knowledge is infused by noise; it signifies that knowledge can be represented in a non-absolute way, with some degree of uncertainty. The Bayesian Belief Network described by Judea Pearl in 1984 is an example of such a probabilistic model. In probabilistic models, concepts are no longer regarded as deterministic sets but rather probabilistic ensembles. Two concepts may, for example, overlap in the set of instances they define. There are two regimes of probabilistic graphical models: flat and hierarchical. The flat regime includes descriptive or declarative models, such as constraint-satisfaction models, Markov random fields, Gibbs models, and Julesz ensembles. The hierarchical regime



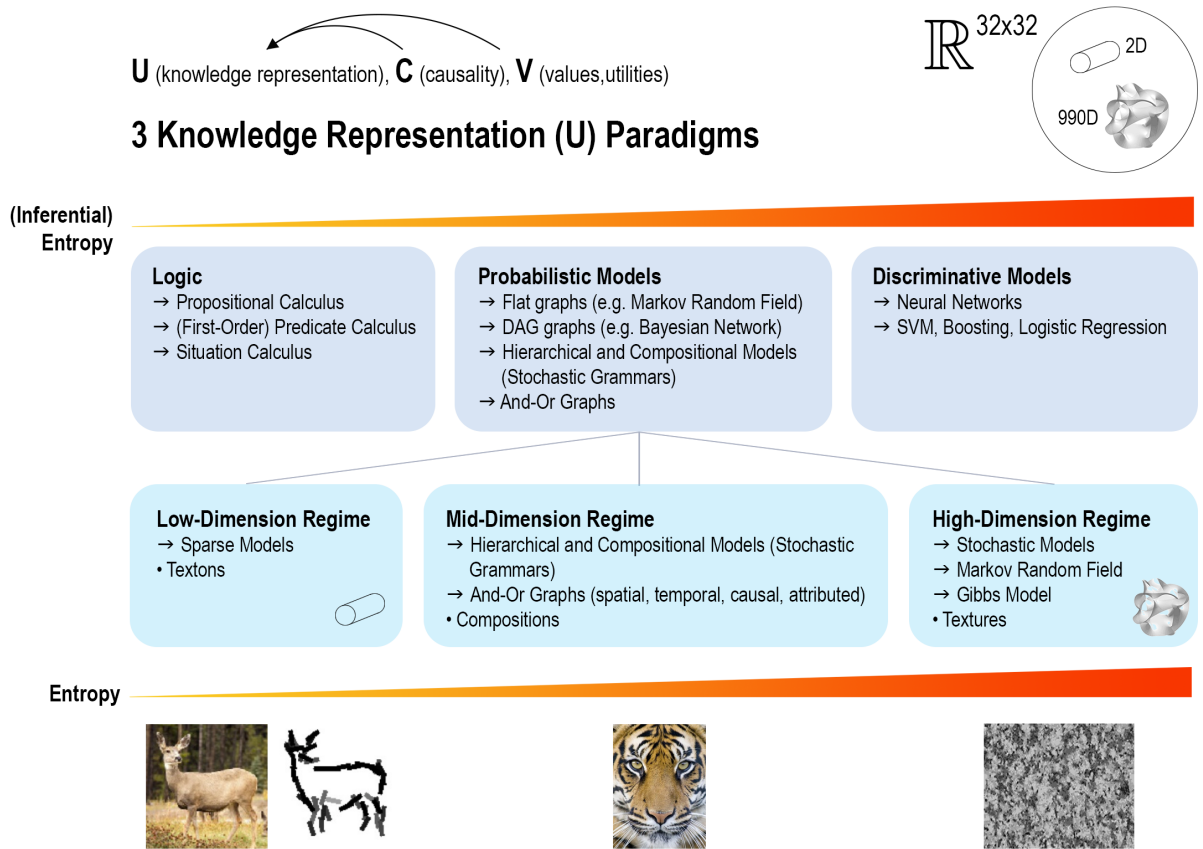


Figure 1.9: At the highest level of vision and cognition, one can discriminate between three types of models: Knowledge Representation (U) models, Causal (C) models, and Value and Utility (V) models. The last two types are discussed in the third book in the series, along with many other more advanced artificial intelligence concepts. Knowledge representation models aim to describe the world as intuitively and as accurately as possible. Causal models represent cause and effect in the physical world. Value models compute how much value is placed on certain objects or ideas; they involve morality. Causal (C) and Value (V) models together guide the creation of the world. For example, chairs are created because they have a certain value (V) and satisfy some causality (C) in physics, allowing an individual to rest when he or she sits down. Causal and Value models, as they uncover knowledge, feed Knowledge Representation models. There are 3 paradigms for knowledge representation: logic models, probabilistic models, and discriminative models. Probabilistic models have low- (e.g., textons), mid-, and high-dimension (e.g., textures) regimes, each increasing in entropy. At the top-right, one can see an  $32 \times 32$  image space, i.e., 1024 dimensions, in which part of the space is only 2D, while another part of the space is 990D. This conveys that dimensionality often varies greatly in just one image, lending instances of each of the low-, mid-, and high-dimension regimes.

includes generative or compositional models, such as Markov trees, stochastic context-free grammars, and sparse coding models. Integrating context into hierarchical models, there are models such as primal sketch, And-Or graphs (spatial, temporal, causal, and attributed), and stochastic context-sensitive grammars. All of these are reviewed in this book.

The last paradigm for knowledge representation in vision, to go along with logic and probabilistic mod-

eling, is discriminative modeling. Discriminative models include neural networks (e.g., convolutional neural networks, recurrent neural networks), boosting, logistic regression, and support vector machines. Discriminative models often have different pipelines for each task in a problem, such as pipelines for object classification, pose estimation, and attribute recognition. This actually leads to non-unified knowledge representation, which is unhelpful for the aim of this book. So, although discriminative models can be used to represent knowledge, they are often used in such a way that does not contribute to a unified representation. Discriminative models also require a huge number of parameters (on the order of  $O(10^7)$ ) and a very large amount of data for supervised training (on the order of  $O(10^6)$ ). Generally, amongst the three paradigms of knowledge representation, discriminative models are very complex while logic is simple; probabilistic models lie somewhere in between, simple enough to understand thoroughly and complex enough to capture complex data distributions. A goal of this book is to integrate these three paradigms providing knowledge representation: logic models, probabilistic models, and discriminative models.

## 1.4 Pursuit of Statistical Models

So, broadly speaking, why are statistical models useful in computer vision, and what is the origin of these models? Some assume that statistical models and probability have a role in computer vision primarily due to the noise and distortion present in natural images. This is truly a misunderstanding. With the abundance of high-quality cameras that exist nowadays, there is rarely a considerable amount of noise or distortion in images anymore. Rather, probability and statistical models actually help capture more detail in images, as opposed to helping represent less detail, i.e., noise or distortion. Accordingly, statistical models, which provide an intrinsic representation of visual knowledge, actually help capture image *regularities*.

As discussed previously, a population of natural image patches can be viewed as a cloud of points in the image space. The distribution of the cloud of points can be described by a probability density function. Image patches of a certain pattern, such as a face, form a subpopulation in this cloud of points. If one randomly samples an image patch from this face subpopulation, then the density function one assigns to the random image patch, naturally, should be the density function of the subpopulation from which it was sampled. In this way, each pattern, or concept, corresponds to a probability distribution defining a subset of the image space, and it can be defined by a probability density function or a statistical model.

In the literature, Grenander (1970) [76] and Fu (1973) [63] pioneered using statistical models for various visual patterns. In the late 1980s and early 1990s, statistical models become popular and then indispensable when the computer vision community recognized that problems, typically shape-from-X problems, are intrinsically ill-posed. If interested, the reader is invited to explore the historical context more. Nowadays, it is known that extra information is needed to account for the regularities in natural images, and statistical models can help to encode or represent these regularities. Crucially, statistical models also assist in learning and recognizing patterns or objects in the first place.

Fig. 1.10 illustrates two methods of pursuing a statistical model of a concept, i.e., of learning a concept  $f$  by approaching  $\Omega_f$  with a sequence of models. In the first method, starting from the entire image space, at each step a new constraint is added to shrink the image space. With more constraints continually added,  $\Omega_f$  is captured from the outside. This method of pursuing a statistical model characterizes descriptive or declarative models, which are flat. Descriptive models represent one regime of probabilistic graphical models and include constraint-satisfaction models, Markov random fields, Gibbs models, Julesz ensembles, and other contextual models. In the second method, starting from a single point or small ball inside  $\Omega_f$ , at each step some dimensions are expanded to gradually fill in  $\Omega_f$  from the inside. This method of pursuing a statistical model characterizes generative or compositional models, which are hierarchical. Generative or compositional models represent another regime of probabilistic graphical models and include Markov

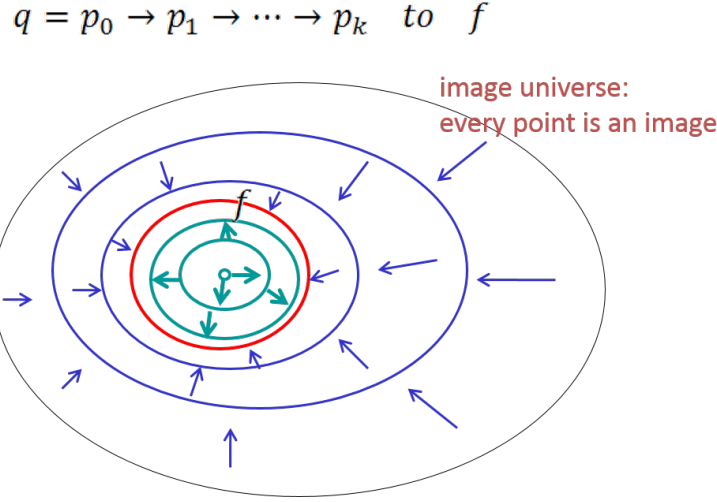


Figure 1.10: This illustrates two strategies of pursuing a statistical model for a concept. First, in descriptive or declarative models, a set  $\Omega_f$  can be pursued through a sequence of models that continually reduce the entire image space to capture  $\Omega_f$  from the outside. The image space is gradually reduced by adding constraints. Second, in generative or compositional models, a set  $\Omega_f$  can be pursued through a sequence of models that continually expand from a single point or small ball to gradually fill in  $\Omega_f$  from the inside. In this case, the image space is gradually expanded by adding dimensions.

trees, stochastic context-free grammars, and sparse coding models. Integrating contextual and hierarchical information are, e.g., primal sketch models, And-Or graphs, and stochastic context-sensitive grammars.

The reason for choosing one of the two model pursuit strategies, descriptive or generative, for a given problem is intuitive. Some sets  $\Omega$  may be of very high dimensionality, and hence it is more efficient to capture them with a descriptive model using constraints, gradually reducing the volume of the search space. These sets  $\Omega$  are modeled through description, hence the name “descriptive” models. Other sets  $\Omega$  may be of much lower dimensionality, and hence it will be more efficient to capture them with a generative model using expansion, gradually increasing the volume of the search space by adding dimensions. These sets  $\Omega$  are modeled by pursuing latent variables, which are meant to encode and explain training examples and may be used to generate new examples, hence the name “generative” models. Some sets  $\Omega$  may be of more middle dimensionality and hence require a more complex combination of a generative and descriptive model.

By analogy, if a teacher is grading a final exam which has a full score of 100, for a very strong student the teacher may start from 100 points and subtract points occasionally for incorrect answers. For a very weak student, the teacher may start from 0 and add points occasionally for correct answers. Accordingly, students at both ends of the spectrum are easy to grade, but students near the exam average require more work. Images for most vision problems are near the middle of an analogous spectrum, making it necessary to capture challenging sets  $\Omega$  that form complex, multimodal distributions of varying dimensionality.

Fig. 1.11 illustrates use cases for the different regimes of models, using only two images, one of red maple leaves and the other of green leaves. Both images are simply shown at five varying levels of magnification. On the very left of the figure, the low-dimension regime may be observed. As the images are of low dimension, it is easiest to generate or compose them through expansion with a generative model, increasing the volume of the search space by adding dimensions. On the very right of the figure, the high-dimension regime may be observed. As the images are of high dimension, it is easiest to describe them with a descrip-



Figure 1.11: Scanning from left to right, low-dimension, mid-dimension, and high-dimension regimes may be observed for two images. There is a gradual transition from images that should be fully modeled by a generative model, on the very left, to images that should be fully modeled by a descriptive model, on the very right.

tive model, reducing the volume of the search space by introducing constraints. From the left to the right of the figure, there is a gradual transition from images that should be fully modeled by a generative model to images that should be fully modeled by a descriptive model. In between, a combination of the two should be used to optimally model the leaves. This is a powerful example because it shows that, even for the same picture but at varying levels of magnification, there is a need for generative and descriptive models and their combination.

Thus far, relatively small image patches have been discussed, e.g.,  $10 \times 10$  pixels. The concepts that arise from small image patches may be considered “atomic concepts,” i.e., the simplest symbols, or descriptors, at the lowest layer of perception. Atomic concepts can be further composed into larger patterns and more abstract concepts. This leads to a hierarchy of concepts at multiple layers, as illustrated by Fig. 1.12.

At the bottom layer, there are concepts for image patches. High-entropy concepts can be described by, e.g., Markov random fields, and low-entropy concepts can be described by, e.g., sparse coding models. At the top layer, there is reasoning based on logic and event calculus. In the middle layers, there are recognition and cognition, which can be described by stochastic grammars allowing for recursive compositions. As will be discussed, stochastic grammars in the form of Spatial-Temporal-Causal And-Or graphs are actually well-suited to learn and describe concepts at all layers.

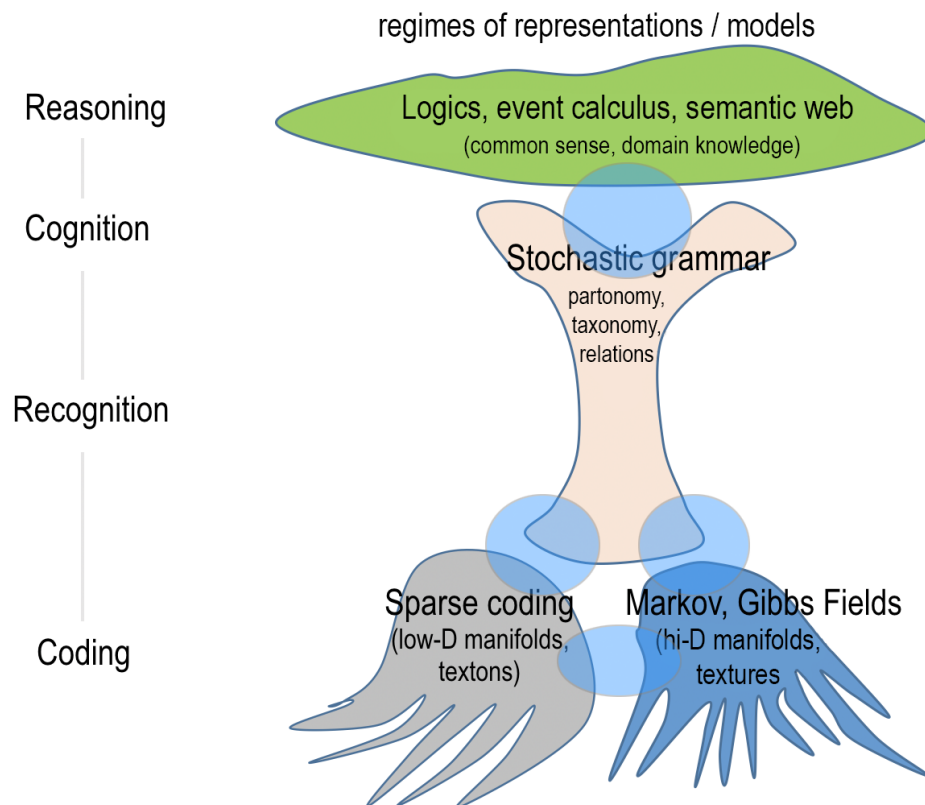


Figure 1.12: Concepts become increasingly abstract, moving up through the hierarchy from coding to recognition to cognition to reasoning. However, concepts at all levels can be learned and described by Spatial-Temporal-Causal And-Or graphs.



## 2

# Statistics of Natural Images

The set of natural images may be defined as the images of the natural world that have been observed by humans. Images and videos captured by humans by cameras in modern civilization represent only a low-resolution, very small subset of this set. Accordingly, the volume of the natural image space far exceeds the volume of the space of images captured by humans, although it is much smaller in comparison to the volume of the entire, unconstrained image space. Natural images contain an overwhelming variety of structures and patterns resulting from a myriad of physical processes. They exhibit hierarchical compositions of objects at a broad continuum of scales. In regard to a captured image, the optical axis is usually, but not always, horizontal.

It may be surprising to note that natural images, with their diverse array of patterns and orientations, consistently share any statistical features. Indeed, they do and so much so that they can be easily distinguished from non-natural images based solely on these features. For example, most natural images contain simple subcomponents such as flat surfaces, well-defined edges, and regular textures. The common appearance of these structures suggests that concise but flexible models can efficiently describe natural images at the most basic level by repeated instances of similar patterns.

Barlow in 1961 [9] and Gibson in 1966 [72] were among the earliest researchers to emphasize the role of ecology in visual perception, i.e., that visual concepts are learned and mentally categorized based on examples in the environment. Computer vision, like human vision, consists of an agent that learns abstract representations from many observed examples. In a computer vision problem, the training data is considered to be and as testing data what would be future observed images by a human. The learned representations of vision models, similar to concepts for humans, should be sufficiently generalizable so that the model can adapt to new, unseen examples just as a human can.

Throughout this book, statistical phenomena associated with natural images are discussed. This begins the process of establishing a common vocabulary and grammar for vision supported by a unified mathematical foundation. In this particular chapter, some initial statistical properties discovered for natural images are examined, such as the  $1/f$ -power law, high kurtosis, and scale invariance. Manifolds in the image universe, image scaling in the image space, and imperceptibility are discussed. There are many motivations for understanding natural image statistics, but a few include potential optimizations in image/video compression due to redundancy reduction and improved image/video coding, an understanding of ecologic influences on neural receptive fields (e.g., how neurons in visual cortical areas *adapted* to visual environments over time), and the knowledge to exploit image regularities and prior models to solve ill-posed problems, such as image restoration (e.g., denoising, inpainting), estimating surface from stereo, motion, texture, etc., and the development of concepts and generative models (as prior knowledge) for scenes, objects, actions, events, and causality from images and video.

## 2.1 Image Space and Distribution

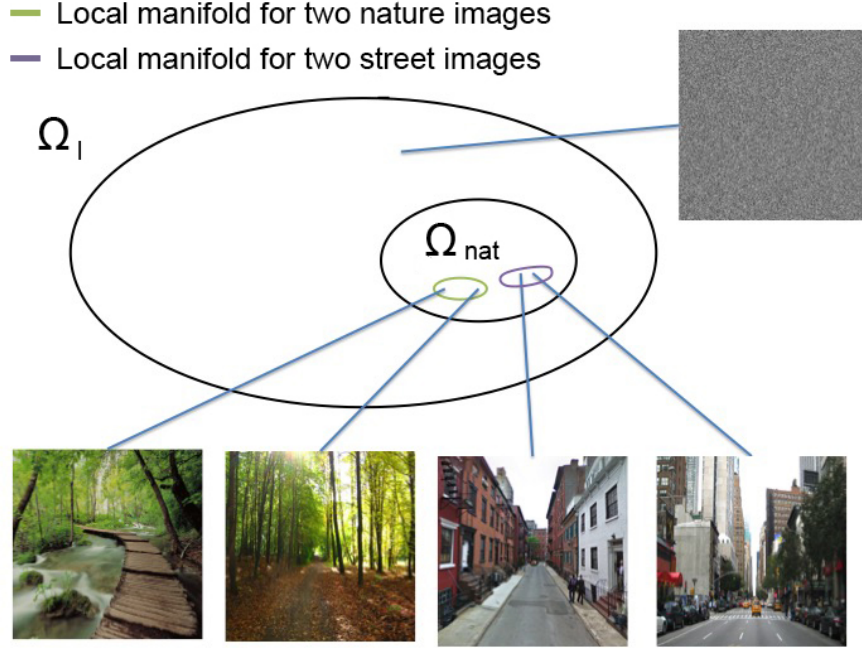


Figure 2.1: The natural image space is a subset of the entire image space, i.e.,  $\Omega_{nat} \subset \Omega_I$ . Lines are drawn from the four natural images below to their positions in  $\Omega_{nat}$ , a high-dimensional space. Similar images lie on local manifolds of  $\Omega_{nat}$ , separated by relatively small geodesic distances. The image of noise does not belong to the natural image space,  $\Omega_{nat}$ .

Beginning with a concrete description of an image space, consider the population of all grayscale image patches of size  $10 \times 10$  pixels with each pixel intensity in the range  $[0, 1]$ . Each possible image patch is a point in a high-dimensional space  $[0, 1]^{100}$ . This means that an image can be represented by a single vector with 100 elements for the 100 pixel intensities. If the images are color instead of grayscale, each of the three color channels (red, green, blue) has its own  $10 \times 10$  image patch and, accordingly, the image space is  $[0, 1]^{300}$ . The population of *natural* image patches forms a cloud of points, which is only a subset of the entire image space. The distribution of this cloud of points exhibits a high-dimensional geometry reflecting the various structures in natural images.

The distribution of natural images, i.e., the cloud of points, can be described by a probability density function, indicating the density of points at each position in the image space. The density at a position in the image space indicates how likely it is an image from that position would be chosen, if it were sampled from the distribution of natural images. The density of the distribution is very uneven. In some positions of the image space, e.g., positions with image patches of forest scenes, the density is high, while in other positions, e.g., one with a random sample from a high-dimensional uniform distribution, the density is virtually zero. Simply put, this is because amongst natural images, image patches of a forest scene are relatively common, whereas an image patch from a high-dimensional, and hence inevitably unique, distribution are less common. The concentration of the density function around certain geometric regions is analogous to the distribution of mass in the universe. The density of mass is high at a star and low across nebulae, clouds of gas and dust in space. And the density is practically zero over large spans of space.



Now some notation should be introduced. Let  $\mathbf{I}(x)$  be an image patch defined on a square or rectangular domain  $\mathcal{D}$ , where  $x \in \mathcal{D}$  indexes the location within the image. The domain, or coordinate space,  $\mathcal{D}$  for location  $x$  can be continuous or discrete. An example of a continuous domain is  $\mathcal{D} = [0, 1] \times [0, 2]$ , which denotes a rectangular  $1 \times 2$  grayscale image with continuous axes. The coordinates for location  $x$  are continuous, i.e.,  $x = ([0, 1], [0, 2])$ . An example of a discrete domain is  $\mathcal{D} = \{1, \dots, 10\} \times \{1, \dots, 10\}$ , which denotes a  $10 \times 10$  pixel grayscale image with discrete axes. In this case, the image is composed of pixels. The coordinates for location  $x$  are discrete, i.e.,  $x = (\{1, \dots, 10\}, \{1, \dots, 10\})$ . The coordinate space  $\mathcal{D}_{\text{color}}$  of a three-channel color image is simply the Cartesian product of the grayscale coordinate space  $\mathcal{D}_{\text{gray}}$  and the set  $\{1, 2, 3\}$  that indexes the color channel, i.e.,  $\mathcal{D}_{\text{color}} = \mathcal{D}_{\text{gray}} \times \{1, 2, 3\}$ .  $\mathbf{I}(x)$  may be treated as a function  $\mathbf{I} : \mathcal{D} \rightarrow \mathcal{V}$ , which gives the intensity values of each pixel in  $\mathbf{I}$ . For continuous pixel intensities between 0 and 1, pixel values  $\mathcal{V} = [0, 1]$ , and for discrete integer-valued pixel intensities between 0 and 255, pixel values  $\mathcal{V} = \{0, \dots, 255\}$ .

When the coordinate space  $\mathcal{D}$  is discrete and a finite set, i.e., for pixel images with a finite number of pixels,  $\mathbf{I}$  may be treated as a vector if an ordering for the pixels is fixed. As such, each  $\mathbf{I}$  becomes a point in the image space  $\Omega_{\mathbf{I}} = \mathcal{V}^{|\mathcal{D}|}$ , in which  $|\mathcal{D}|$  is the total number of pixels in the image. (For color images, each channel has its own separate pixels.) For example, every  $10 \times 10$  pixel grayscale image patch  $\mathbf{I}$ , with pixel intensities between 0 and 1, is a point  $\mathbf{I} \in \Omega_{\mathbf{I}} = [0, 1]^{100}$ . (The notation  $\Omega_{\mathbf{I}} = [0, 1]^{100}$  indicates that the sample space  $\Omega_{\mathbf{I}}$  is made up of samples with 100 pixels that take a continuous value between 0 and 1.) Similarly, every  $10 \times 10$  pixel color image, with pixel intensities for each channel between 0 and 1, is a point  $\mathbf{I} \in \Omega_{\mathbf{I}} = [0, 1]^{100 \times 3} = [0, 1]^{300}$ . A more technical definition of  $\Omega_{\mathbf{I}}$  is needed for the cases in which  $\mathcal{D}$  is an infinite set, but for the majority of topics covered in this book, this discrete, finite, pixel-based description is sufficient.

Now consider the cloud of points  $\Omega_{\text{nat}} \subset \Omega_{\mathbf{I}}$  made up of *natural* image patches. The distribution of  $\Omega_{\text{nat}}$  can be described by a density function  $f : \Omega_{\mathbf{I}} \rightarrow \mathbb{R}$ , such that  $f \geq 0$  and  $\int_{\Omega_{\mathbf{I}}} f(\mathbf{I}) d\mathbf{I} = 1$ . Intuitively, for each point  $\mathbf{I} \in \Omega_{\mathbf{I}}$ , let  $\mathcal{N}(\mathbf{I})$  be a small neighborhood around  $\mathbf{I}$ . Then  $f(\mathbf{I})$  is the limit of the proportion of points in the population that belong to  $\mathcal{N}(\mathbf{I})$  divided by the volume of  $\mathcal{N}(\mathbf{I})$ , as the volume of  $\mathcal{N}(\mathbf{I}) \rightarrow 0$ . This is simply the asymptotic extension of the common notion of population density. For instance, the population density of Los Angeles is the number of people in Los Angeles divided by the area of Los Angeles.

In the explanation above, there is nothing probabilistic. The density function  $f$  is simply a deterministic description of a cloud of points in a high-dimensional space. Now, however, the probability is introduced. If an image patch  $\mathbf{I}$  is randomly sampled from the population of natural image patches, i.e., a point from the cloud of points described by density function  $f(\mathbf{I})$  is randomly sampled, then image patch  $\mathbf{I}$  is said to be a random sample, or example, from  $f(\mathbf{I})$ . Accordingly,  $f(\mathbf{I})$  becomes the density function for the random image patch  $\mathbf{I}$ , and  $\mathbf{I} \sim f(\mathbf{I})$  is written. The goal of learning, then, is to gain some knowledge about the density function  $f(\mathbf{I})$  based on a set of examples  $\{\mathbf{I}_1, \dots, \mathbf{I}_M\}$  that are sampled from  $f(\mathbf{I})$  independently.

## 2.2 Information and Encoding

Recall that concepts (in the model space) are represented by, or correspond to, sets (in the image space). They can be defined by a distribution or density function  $f$  focused on their corresponding set. A concept can be general, such as the concept of natural images, a certain texture, or white noise, or it can be more specific, such as the concept of the face of a tiger. In any case, a concept is represented by a set  $\Omega_f$  in the image space  $\Omega_{\mathbf{I}}$ . Images from the density function, i.e.,  $\mathbf{I} \sim f(\mathbf{I})$ , are focused on the subspace  $\Omega_f \subseteq \Omega_{\mathbf{I}}$ .

In high-dimensional image spaces, common to vision problems, a notable phenomenon occurs. The density function  $f(\mathbf{I})$  is close to zero outside of  $\Omega_f$ . It is nearly uniform inside of  $\Omega_f$ , but in the rest of the

image space  $\Omega_{\mathbf{I}}$ , it is close to zero. So it can be roughly stated that  $f(\mathbf{I}) = \frac{1}{|\Omega_f|}$  for  $\mathbf{I} \in \Omega_f$  and  $f(\mathbf{I}) = 0$  for  $\mathbf{I} \notin \Omega_f$ . The high-dimensional nature of the data in many vision problems makes it such that the image space  $\Omega_{\mathbf{I}}$  is largely dominated by white noise. As a result, a nearly uniform density exists over  $\Omega_{\mathbf{I}}$ , as roughly 99.9% of it is simply white noise. The goal of learning a concept, then, is to identify which set  $\Omega_f$  in the largely meaningless, noise-dominated set  $\Omega_{\mathbf{I}}$  corresponds to the given concept, i.e., find a probability density function  $f$  defining it.

Now consider two models, an "unfocused" model  $g(\mathbf{I})$  that has a uniform density within  $\Omega_{\mathbf{I}}$  and a "focused" model  $f(\mathbf{I})$  that has a uniform density within the subset of natural images  $\Omega_{\text{nat}}$  and is approximately zero outside this set. The volume of the subset  $\Omega_{\text{nat}}$  can be described using *entropy*, which is the average rate information is produced by a stochastic form of data. Entropy can be intuitively understood as a measure of disorder. Deterministic variables (with perfect order) possess zero entropy, while uniform random variables (with the perfect disorder) possess the maximum entropy of all possible distributions over the support of the uniform variable.

The fact that the volume of the subset  $\Omega_{\text{nat}}$  can be described using *entropy* is an important point. It means the complexity of a concept  $f$  in the model space can ultimately be expressed by the entropy, i.e., the log volume, of its corresponding set  $\Omega_f$  in the image space. The entropy of an image set, by giving the disorder of pixel information of that set, encodes the complexity of the corresponding concept. In computer vision, entropy and entropy rate are both measured in pixels.

The entropy  $H$  of a distribution  $p(\mathbf{I})$  over a finite image space  $\Omega_{\mathbf{I}}$  is

$$H(p) := - \sum_{\mathbf{I} \in \Omega_{\mathbf{I}}} p(\mathbf{I}) \log_2 p(\mathbf{I}).$$

The use of  $\log_2$  gives a measurement of entropy in terms of bits but, in general, any base can be used for the logarithm, as different bases only scale  $H$  differently. For a distribution  $q$  that is uniform over a set  $\Omega_q \subseteq \Omega_{\mathbf{I}}$  and zero elsewhere, the entropy is

$$H(q) = - \sum_{\mathbf{I} \in \Omega_q} \frac{1}{|\Omega_q|} \log_2 \frac{1}{|\Omega_q|} = \log_2 |\Omega_q|,$$

in which the sum over  $\Omega_{\mathbf{I}}$  reduces to a sum over  $\Omega_q$ , as density is zero outside of  $\Omega_q$ . Therefore, the entropy of a uniform distribution  $q$  is a measure of the log volume of the space  $\Omega_q$ . (Note that this is why the general formulation of entropy has a negative sign, i.e., so the resulting entropy in operations like the one above is positive and not negative.) In particular, for the "unfocused" model that is uniform over the entire image space  $\Omega_{\mathbf{I}}$ ,  $H(f) = \log_2 |\Omega_{\mathbf{I}}|$ , and for the "focused" model that is uniform over only the natural image space  $\Omega_{\text{nat}}$  and zero elsewhere,  $H(g) = \log_2 |\Omega_{\text{nat}}|$ . Later chapters describe a more detailed relationship between entropy and learning using information theory. For now, it is only important to note the connection between entropy and the volume of an image space.

The entropy of a subset of the image space encodes the amount of information needed to represent members of the subset. Consider the space  $\Omega_h$  of grayscale images that have the same fixed-width horizontal black bar against a white background. Each image can be described by a single number that represents the vertical position of the bar. Suppose that there are  $|\Omega_h| = n_h$  possible positions for the horizontal bar. Now consider the space  $\Omega_{hv}$  of grayscale images that have the same fixed-width horizontal black bar and the same fixed-width vertical black bar against a white background. All images from this set can be described by exactly two numbers representing the locations of the two bars. Suppose that there are  $n_h$  positions for the horizontal bar and  $n_v$  positions for the vertical bar, such that  $|\Omega_{hv}| = n_h n_v$ . The amount of encoding information needed in the space of vertical and horizontal bars is higher than that of horizontal bars alone, and this is reflected in the difference  $\log_2 |\Omega_h| - \log_2 |\Omega_{hv}| = -\log_2 n_v < 0$ .

In the case of images with  $|\mathcal{D}|$  pixels, e.g.,  $|\mathcal{D}| = n^2$  for a grayscale image with  $n$  pixels for both its height and width, the *entropy rate*  $\bar{H}$  scales entropy relative to the image size, lending a per-pixel entropy. For a density  $q$  uniformly distributed over  $\Omega_q$ , the entropy is

$$\bar{H}(q) = \frac{H(q)}{|\mathcal{D}|} = \frac{1}{|\mathcal{D}|} \log_2 |\Omega_q|.$$

For example, consider the space of  $n \times n$  grayscale images where  $\mathcal{V} = \{0, \dots, 255\}$ , indicating each pixel value is an integer from 0 to 255. In this case, there is the image space  $\Omega_{\mathbf{I}} = \{0, \dots, 255\}^{n^2}$  and the log volume of the image space  $|\Omega_{\mathbf{I}}| = 256^{n^2}$ . Therefore, the unfocused model  $g(\mathbf{I})$  with uniform density in  $\Omega_{\mathbf{I}}$  has entropy rate

$$\bar{H}(g) = \frac{1}{n^2} \log_2 |\Omega_{\mathbf{I}}| = \frac{n^2 \log_2 256}{n^2} = 8 \text{ bits/pixel}.$$

This means that an image randomly sampled from  $\Omega_{\mathbf{I}}$  can be encoded using an average of 8 bits per pixel.

Now, considering a natural image  $I \in \Omega_{\text{nat}}$  sampled from  $f$ , the many regularities found in natural images allow for a drastic reduction in the amount of bits needed for encoding the image. The empirical upper bound for the average bits per pixel, i.e., the entropy rate, needed to encode natural images is 0.3:

$$\bar{H}(f) = \frac{1}{|\mathcal{D}|} \log_2 |\Omega_{\text{nat}}| \leq 0.3 \text{ bits/pixel}.$$

For the unfocused model  $g(\mathbf{I})$  with uniform density in  $\Omega_{\mathbf{I}}$ , 8 bits per pixel are needed to encode any potential image that could be drawn from  $\Omega_{\mathbf{I}}$ ; it is not possible to compress uniform random samples drawn from the entire image space  $\Omega_{\mathbf{I}}$ . However, for the unfocused model  $f(\mathbf{I})$  with uniform density within  $\Omega_{\text{nat}}$  and zero density elsewhere, the reduction in entropy rate from 8 bits/pixel to 0.3 bits/pixel allows for compression. This decreased entropy rate of 0.3 bits/pixel gives an intuitive measure for the amount of compression possible when the regular features found across previously observed natural images are used to encode new images.

Here the first statistical observation for natural images may be noted: redundancy. The redundancy in real world images allows for continual improvements to image/video compression. Instead of needing 8 bits to encode a pixel, only an upper bound of 0.3 bits is needed. Using any more bits than this to encode a pixel would not take advantage of all the redundancies in natural images. Note that compression methods work well on natural images but poorly, for example, on a noisy image such as a QR code (a matrix barcode).

Now a rough estimate of the size of the observed natural image space  $\Omega_{\text{nat}}$  may be given. Assume there are 10 billion people who have lived on Earth, and each person lived or will live to be 100 years old, and each person observed 20 images per second. The volume of the natural image space,  $|\Omega_{\text{nat}}|$ , as seen by humans, maybe estimated as

$$|\Omega_{\text{nat}}| \approx 10^7 \text{ humans} \times 100 \text{ years} \times 365 \text{ days} \times 24 \text{ hours} \times 3600 \text{ sec} \times 20 \text{ fps} \approx 6.3 \times 10^{22} \text{ images}.$$

The size of the total image universe  $|\Omega_{\mathbf{I}}| = 256^{n^2}$  discussed previously is much larger than  $|\Omega_{\text{nat}}|$ , even for very small image spaces, e.g., image patches with  $n = 10$  pixels for both height and width. Keep in mind that the models in this book are meant to represent samples from the natural image space  $\Omega_{\text{nat}}$  and reflect the regularities observed therein.

## 2.3 Image Statistics and Power Law

The universe is composed of an ensemble of structures at many different scales. Using a coarse scale, a structure may be perceived as an atomic entity, while using a finer scale, the same structure may be perceived

as a compound entity with many different parts. Structured ensembles that can be viewed as a singular entity are commonly referred to as objects. In natural images, objects can be identified by regularities (such as consistent color or texture or connected surfaces) that enable perception of them as distinct concepts. The regularities observed in natural images reflect the hierarchically structured order of the universe. Noise images, on the other hand, do not obey such laws. As a result, natural images and noise can be distinguished based on numerical summaries of structural regularities. These statistical properties form the foundation for a more detailed understanding of images pursued in later chapters.

In natural images, nearby pixel intensities are typically similar in value, but occasional jumps or discontinuities can be observed as well. As two neighboring cities are not usually expected to have a large difference in elevation, two neighboring pixels are not usually expected to have a large difference in intensity, i.e., pixel values, or two neighboring patches to have a large difference in smoothness. This describes regions of low power and high abundance in the landscape, as they are common but low energy relative to other regions. However, if a city is built near a mountain range such as the great city of Kathmandu, a large increase in elevation between Kathmandu and the nearby Mt. Everest base camp is indeed observed, which is analogous to neighboring pixels having a large difference in intensity, indicating the potential presence of an edge or some strong feature. This describes regions of high power and low abundance in the landscape, as they are not as common but exhibit high energy relatively. Considering the entire world as a landscape, regions encompassing cities such as Kathmandu have high power but are not very abundant. Most regions in the world have relatively consistent elevation, i.e., low power, and natural image landscapes follow the same trend. Lastly, the elevation of two distant cities, e.g., Kathmandu and Los Angeles, is likely to be very different, just as the intensity of two distant pixels.

A simple measure of similarity between the intensities of two pixels is covariance or correlation. For an image  $\mathbf{I} \in \mathcal{V}^{|\mathcal{D}|} \subset \mathbb{R}^{|\mathcal{D}|}$  with  $|\mathcal{D}|$  pixels, assume that the marginal mean is normalized to be 0 and the marginal variance to be 1, i.e.,  $\frac{1}{|\mathcal{D}|} \sum_x \mathbf{I}(x) = 0$  and  $\frac{1}{|\mathcal{D}|} \sum_x \mathbf{I}(x)^2 = 1$ . Then the covariance between pixel  $x = (x_1, x_2)$  and  $x + d = (x_1 + d_1, x_2 + d_2)$  can be calculated as  $C(d) = \frac{1}{|\mathcal{D}|} \sum_x \mathbf{I}(x)\mathbf{I}(x + d)$ . The covariance  $C(d)$  is a function of the distance between pixels,  $d$ , and can be large when  $|d| = \sqrt{d_1^2 + d_2^2}$  is small and decays to 0 as  $|d|$  becomes large. An equivalent measure of the similarity between the intensities of two pixels is the power spectrum, or power spectral density, of the image in the Fourier domain. The power spectrum of the image,  $A^2(f)$ , is simply the Fourier transform of the covariance function  $C(d)$ , so the covariance and the power spectrum are simply two ways of capturing the same statistical property: the second-order moment of the image distribution.

The power spectrum  $A^2(f)$  is the square of the amplitude spectrum  $A(f)$ . Recall that neighboring pixels exhibit relatively high power, and hence high amplitude, for large differences in pixel intensities and vice-versa. The power spectrum provides a way to capture and represent these differences. It describes how relations between image pixels are distributed for different frequencies, i.e., how the image content fluctuates at different frequencies, or scales. In general, the power spectrum describes the distribution of power into frequency components composing a given signal. As the relevant application is vision, the signal is an image and the frequency is spatial, not temporal. Spatial frequency is periodic across positions in space and a measure of how often sinusoidal components (as determined by the Fourier transform) of the structure repeat per unit of distance, in this case pixels. According to Fourier analysis, any physical signal can be composed into a number of discrete frequencies, or a spectrum of frequencies over a continuous range. Accordingly, images may be composed as a number of discrete spatial frequencies. The statistical average of a signal as analyzed in terms of its frequency content is called its spectrum, and summation or integration of the spectral components yields the total power (for a physical process) or variance (for a statistical process). The term from physics will be used: power.

As mentioned previously, it may be surprising to note that natural images, with their diverse patterns and

orientations, share any consistent statistical features at all, but indeed they do. Natural images can easily be distinguished from non-natural images, such as images of random patterns for example, by their amplitude spectra or power spectra. In images of random patterns, such as white noise, the amplitude spectra is, by definition, flat. Natural images, however, display the greatest amplitude at low frequencies (at the center of the plot in Fig. 2.2) and decreasing amplitude as the frequency increases, regardless of the orientation of the image, as depicted in Fig. 2.2.

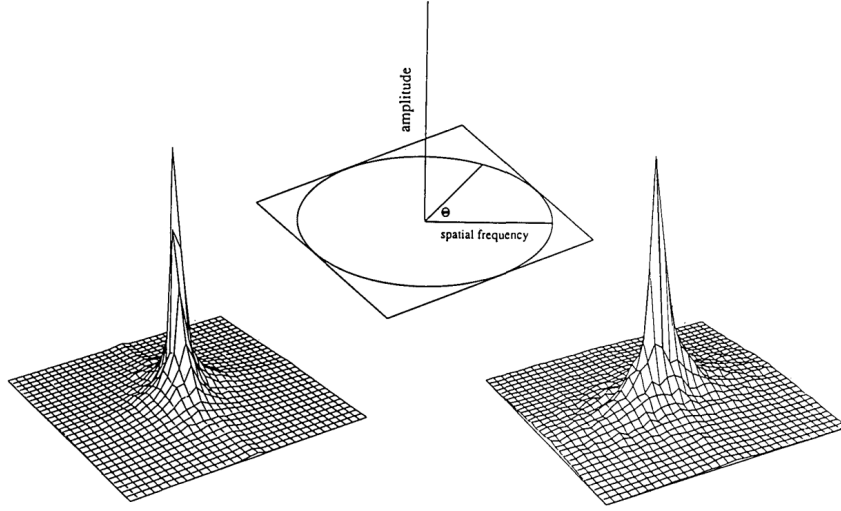


Figure 2.2: From Field (1987) [51], two natural images display similar amplitude spectra. The center of the plots represents 0 spatial frequency, and frequency increases as a function of distance from the center. Orientation is represented by the angle from the horizontal. Note how amplitude decreases sharply with increasing frequency at all orientations. To be precise, amplitude decreases by a factor of  $1/f$ , and power decreases by a factor of  $1/f^2$ , in which  $f$  is frequency [51].

The Fourier transform often decomposes a function of time, i.e., a signal, in the time domain into its constituent temporal frequencies in the frequency domain. In this case, it decomposes a function of space, i.e., an image, in the spatial domain into its constituent spatial frequencies in the frequency domain. It is a linear transform that projects an image, in the form of a vector, onto an eigenvector space, transforming the image from a Euclidean basis to a Fourier basis. In many cases with discrete signals, the Fourier transform is invertible, and no data is lost if the inverse Fourier Transform is used to recover the original signal. Even when the original signal is real-valued, frequency signals contain both real and imaginary components, so absolute value is used to represent the total power or energy of a signal. A Fast Fourier Transform (FFT) may be performed on an image  $I \in R^{|\mathcal{D}_1| \times |\mathcal{D}_2|}$  using

$$\hat{\mathbf{I}}(\xi, \eta) = \sum_{(x,y) \in \mathcal{D}_1 \times \mathcal{D}_2} \mathbf{I}(x, y) e^{-i2\pi(\frac{x\xi}{H} + \frac{y\eta}{W})} \quad (2.1)$$

in which  $|\mathcal{D}_1| = H$  is the number of pixel rows,  $|\mathcal{D}_2| = W$  is the number of pixel columns, and  $(\xi, \eta)$  represents the horizontal and vertical frequencies. The absolute value of the resulting complex number

$$A(\xi, \eta) = |\hat{\mathbf{I}}(\xi, \eta)|$$

is the Fourier amplitude, which gives the magnitude of the frequency  $(\xi, \eta)$  within the image signal  $\mathbf{I}$ . The Fourier power  $A^2(\xi, \eta)$  is the square of the Fourier amplitude  $A(\xi, \eta)$ . Intuitively, the Fourier power, for

given horizontal and vertical frequencies  $(\xi, \eta)$ , encodes how many components of the image vector are projected onto these frequencies at a given orientation.

An interesting empirical observation for natural images is that, for all frequency  $f = \sqrt{\xi^2 + \eta^2}$ :

$$A(f) \propto 1/f \quad (2.2)$$

$$\log A(f) = \text{constant} - \log f \quad (2.3)$$

Thus, the amplitude  $A$  of a Fourier coefficient is inversely related to the frequency  $f$ . The inverse relationship between amplitude and frequency is referred to as the inverse power law for natural images. This is the second statistical observation for natural images. Fig. 2.3 plots the  $\log A(f)$  over  $\log f$  for the four natural images from Fig. 2.1, and the curves can be fitted by straight lines with a slope close to  $-1$ , showing the inverse relationship. Not all natural images exhibit the inverse power law, e.g., natural images of fields of grass or the night sky, but this can be expected for most natural images [51].

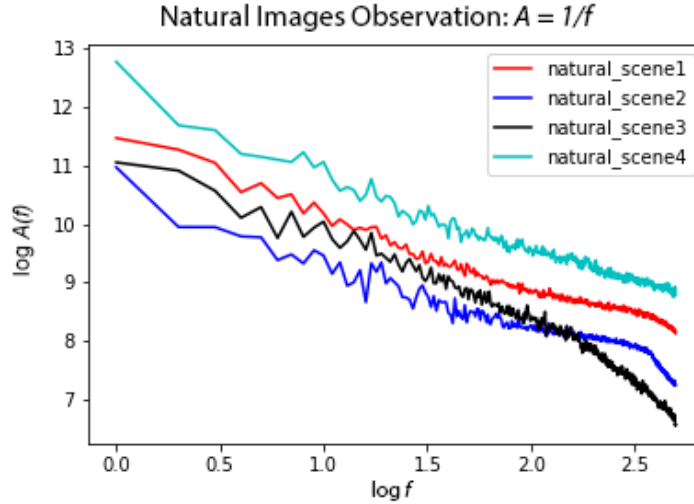


Figure 2.3: Note the log of the amplitude,  $\log A(f)$ , for a given log of the frequency,  $\log f$ , for the four natural images from Fig. 2.1. In natural images, it is observed that the amplitude  $A(f)$  is inversely related to the frequency  $f$ , i.e.,  $A(f) = 1/f$ . Accordingly, the power  $A^2(f) = 1/f^2$ . This property for natural images is referred to as the inverse power law.

The inverse relationship between amplitude and frequency may be used to calculate the total Fourier power  $A^2(f)$  in the frequency band, or octave,  $[f, 2f]$ :

$$\int \int_{f^2 \leq \xi^2 + \eta^2 \leq (2f)^2} |\hat{\mathbf{I}}(\xi, \eta)|^2 d\xi d\eta = 2\pi \int_{f^2}^{4f^2} \frac{1}{f^2} df^2 = \text{constant}, \quad \forall f. \quad (2.4)$$

As depicted in Fig. 2.4, as frequency increases, and hence the total area within each frequency band increases, amplitude decreases. As a result, the power remains constant overall, i.e., the power for a given frequency band  $[f, 2f]$  is equal to the power for frequency bands  $[2f, 4f]$ ,  $[4f, 8f]$ , and so forth. This is what is communicated by Equation 2.4. This observation that natural images contain equal power across frequency bands reveals that they are scale-invariant, lending the same power independent of frequency  $f$ , or scale, i.e., the viewing distance.

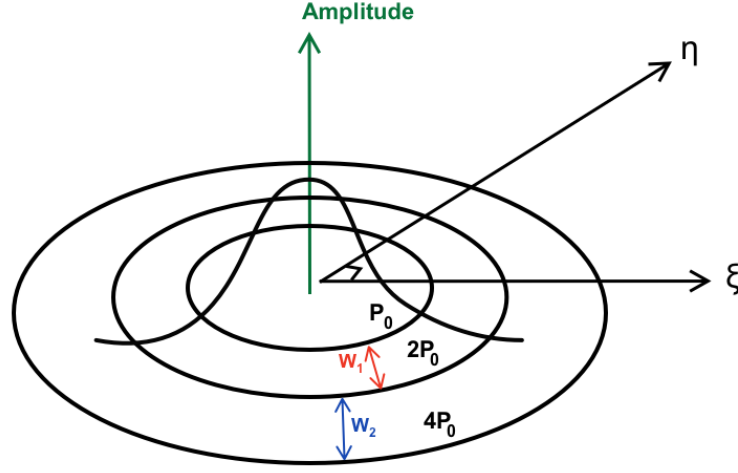


Figure 2.4: Because amplitude decreases as the frequency (and hence the total area within each frequency band) increases, natural images effectively exhibit constant power within each band such as  $\omega_2$ .  $\eta$  and  $\xi$  represent the vertical and horizontal frequencies, respectively.  $2P_o$  indicates a doubling of the frequency of the original image or image distribution,  $P_o$ .

The second-order properties of the image distribution, captured by the covariance  $C(d)$  and the power spectrum  $A^2(f)$ , can be fully reproduced by a multivariate Gaussian distribution with a variance-covariance matrix that agrees with  $C(d)$  or  $A^2(f)$ . For instance, a Gaussian model that accounts for the inverse power law is surprisingly simple. It was shown by Mumford (1995) [161] that a Gaussian Markov Random Field (GMRF) model, as in Equation 2.5, has exactly  $1/f$ -Fourier amplitude:

$$p_{1/f}(\mathbf{I}; \beta) = \frac{1}{Z} \exp \left\{ - \sum_x \beta |\nabla \mathbf{I}(x)|^2 \right\}, \quad (2.5)$$

in which  $|\nabla \mathbf{I}(x)|^2 = (\nabla_x \mathbf{I}(x))^2 + (\nabla_y \mathbf{I}(x))^2$ , and

$$\nabla_x = \mathbf{I}(x_1 + 1, x_2) - \mathbf{I}(x_1, x_2) \quad \text{and} \quad \nabla_y = \mathbf{I}(x_1, x_2 + 1) - \mathbf{I}(x_1, x_2)$$

are discrete approximations of the gradients.

Since  $p_{1/f}(\mathbf{I}; \beta)$  is a Gaussian model, one can easily draw a random sample  $\mathbf{I} \sim p_{1/f}(\mathbf{I}; \beta)$ . Fig. 2.5 shows a typical sample image by Mumford (1995). This model matches the local regularity of natural image statistics and nothing else. It can be considered a natural image model with one constraint, making it only slightly better than a random noise model. Large values of the parameter  $\beta$  lead to samples with large regions of similar intensity, and small values of  $\beta$  lead to samples with more variation in nearby pixels. The limiting case  $\beta \rightarrow \infty$  concentrates all mass on the zero image  $\mathbf{I} = 0$ . From this, it can be concluded that although certain statistics like the covariance and power spectrum found in natural images are also captured by this model, these statistics clearly do not contain sufficient complexity to capture all key features needed for natural image modeling and inference.

The power spectrum can also be pooled by Gabor filters centered at different frequencies. Gabor filters are sine and cosine waves multiplied by elongate Gaussian functions, and the filter response is a localized Fourier transform. Let  $F$  be a Gabor filter whose sine and cosine waves have frequency  $\omega$ . Let  $[F * \mathbf{I}](x)$

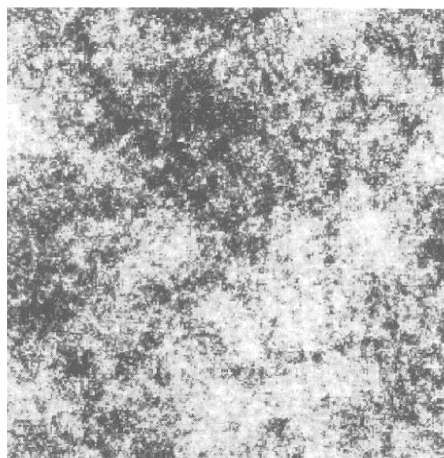


Figure 2.5: Observe a random sample from the Gaussian Markov Random Field model. Note that it has very little structure, a large number of edges, and minimal flatness in its energy landscape. Clearly, second-order properties such as covariance and spectrum do not sufficiently capture key features of natural images.

be the Gabor filter response at location  $x$ . Then,  $[F * \mathbf{I}](x)$  measures the frequency content of  $\mathbf{I}$  around frequency  $\omega$  at location  $x$ . Due to the spatial localization of filter  $F$ ,  $F$  also responds to sine and cosine waves whose frequencies are close to frequency  $\omega$ . Thus, if  $\sum_x |F * \mathbf{I}|^2 / |\mathcal{D}|$  is pooled, it will be the average of the power spectrum within the band of frequencies to which filter  $F$  responds. By the inverse power law,  $\sum_x |F * \mathbf{I}|^2 / |\mathcal{D}|$  remains scale-invariant, i.e., it is constant for different frequencies  $\omega$ .

## 2.4 Kurtosis and Sparsity

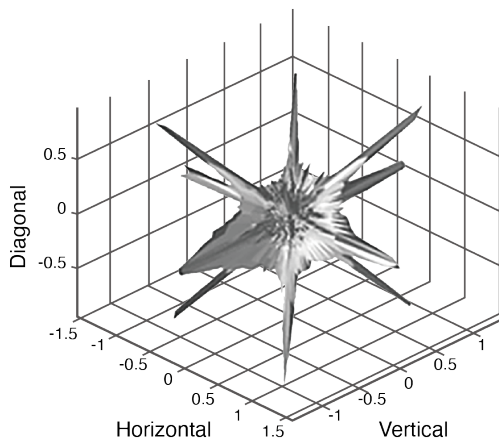


Figure 2.6: The distribution of natural images has many low-dimensional spikes.

As mentioned above, second-order properties such as the covariance, power spectrum, or average of the squared filter responses can be fully reproduced by a multivariate Gaussian distribution with a matching variance-covariance matrix. The Gaussian distribution can be thought of as a cloud of points, which for a Gaussian forms an ellipsoid shape. The ellipsoid-shaped Gaussian distribution is rather dull. It cannot be



expected to capture the presumably highly complex shape of the natural image distribution. For example, the distribution of natural images has many low-dimensional spikes, as illustrated by Fig. 2.6. As a result, researchers have put immense effort into finding patterns of deviation from the Gaussian distribution, attempting to pinpoint non-Gaussian features. The study of natural image statistics has leveraged many properties, from covariances to histograms of filter responses, e.g., Gabor filters. While covariances only measure second-order moments, histograms of filter responses include higher-order information such as skewness and kurtosis.

To define skewness and kurtosis, let  $X$  denote a random variable with mean  $\mu$  and variance  $\sigma^2$ , such that  $X$  can be normalized by the transform  $(X - \mu)/\sigma$ . Skewness is the third-order statistical moment  $E[((X - \mu)/\sigma)^3]$ , and it measures the asymmetry of the distribution. Kurtosis is the fourth-order statistical moment  $E[((X - \mu)/\sigma)^4]$ , and a histogram with heavy tails demonstrates high kurtosis. In measuring skewness and kurtosis of a random variable, the first-order moment  $\mu$  and second-order moment  $\sigma^2$  only serve to normalize the random variable, making its mean equal to zero and variance equal to one.

With histograms, the idea of a 1D marginal projection is utilized in order to gain an understanding of a high-dimensional space such as  $\Omega_{\text{nat}}$ . A histogram can be regarded as a 1D marginal projection because it contains statistics in one dimension, i.e., a line. Histograms allow peaking into a high-dimensional space  $\Omega_f$  and learning about it by learning many 1D marginal densities of  $f(\mathbf{I})$ . By learning these densities, the idea is that a probability density could be constructed to represent  $\Omega_f$  in terms of all of these learned 1D marginal densities of  $f(\mathbf{I})$ .

The use of histograms leads to the third statistical observation about natural images: they exhibit high kurtosis. The histograms of Gabor filter responses to natural images, found by Equation 2.6, are highly kurtotic, i.e., heavy-tailed (Field, 1994). Here,  $F$  is a Gabor filter and  $f_n(I)$  is a "focused" model, which recall has zero density outside of the space of natural images  $\Omega_{\text{nat}}$ . The fact that the histograms have tails heavier than a Gaussian distribution reveals that natural images have high-order, non-Gaussian structures. Gabor filters are primarily used to detect orientation, but other filters are also used such as gradient filters, Haar filters, and so forth.

The histogram of the filter response is calculated by the following equation:

$$h(a) = \int_{\Omega_{\mathbf{I}}} f(\mathbf{I}) \delta(\langle F, \mathbf{I} \rangle - a) d\mathbf{I} \quad (2.6)$$

The histograms defined by Equation 2.6 can also be visualized, as shown in Fig. 2.7. These histograms represent 1D statistical properties for an image  $\mathbf{I}$ , located in  $\Omega_{\text{nat}} \subset \Omega_{\mathbf{I}}$ , a high-dimensional space.  $F$  is a local filter, one that has different responses to different regions of image  $I$  that is represented by the points; in linear algebra, the term "base" is often used instead of filter. The inner product  $\langle F, \mathbf{I} \rangle$  is zero when  $F \perp \mathbf{I}$ . The two filter responses illustrate the binning operation performed in Equation 2.6. High kurtosis is observed due to the vast amount of orthogonality between filter  $F$  and image  $\mathbf{I}$ . Remember that kurtosis is associated with the tails of a distribution and not its peak. Accordingly, it is the greater extremity of deviations, or outliers, displayed in the histograms in Fig. 2.7, not the high peaks of these histograms, that portray the high kurtosis of the natural image. Any filter  $F$  response would show the same highly kurtotic structure.

The high kurtosis in natural images is only marginal evidence for hidden structures in natural scenes. A direct way to discover structures and reduce image redundancy is to transform an image into a superposition of image components. This can be done, for example, with Fourier transforms, wavelet transforms [151], various image pyramids [201] for generic images, and principal component analysis (PCA) for particular ensembles of images, such as face images. The transformation from image pixels to a linear basis, such as a Fourier basis, wavelets basis, or PCA basis, achieves two desirable properties. First, the transformation

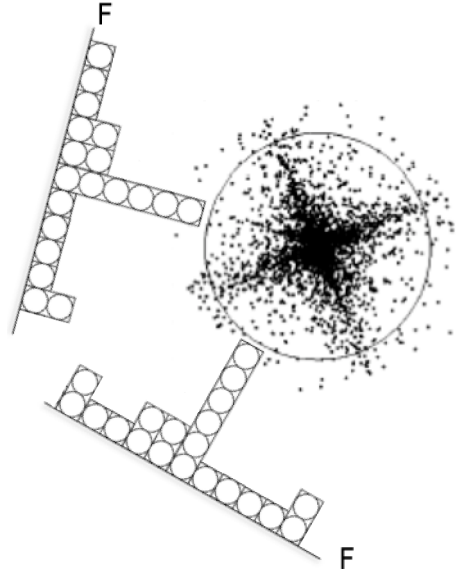


Figure 2.7: Note how the histograms, or 1D marginal projections, which capture the filter responses of a filter  $F$  to a natural image, display high kurtosis. That is, they have a heavy-tailed structure, exhibiting great extremities of deviations, or outliers. An important statistical observation of natural images is that they have a highly kurtotic structure.

induces variable decoupling; coefficients of these bases are less correlated or become independent in an ideal case. Second, the transformation induces dimension reduction; the number of basis vectors needed to approximately reconstruct an image is often ultimately much smaller than the number of pixels.

If one treats an image  $\mathbf{I}(x)$  as a function defined on the domain  $\mathcal{D}$ , then one may also use harmonic analysis to perform image dimension reduction. Harmonic analysis decomposes various classes of functions (i.e., mathematical spaces) or signals into a superposition of basic waves or basis systems. But the population of natural images cannot be fully captured by such functional classes, and so a better solution is needed for image decomposition. From this, an inspiring idea came about: sparse coding with an overcomplete basis, or dictionary, introduced by Olshausen and Field in 1996 [174]. Sparse coding algorithms learn useful sparse representations of data. Given a certain number of dimensions, they learn an overcomplete basis to represent the data. An overcomplete basis signifies that there is redundancy in the basis, as vectors “compete” to represent data more efficiently. This means not all dimensions are needed to represent a data point; some may be set to 0. With an overcomplete basis, an image may be reconstructed by a small, i.e., sparse, number of basis vectors in the dictionary. Olshausen and Field then learned an overcomplete dictionary from many natural images. This often leads to 10 to 100 folds of potential dimension reduction. For example, an image of  $200 \times 200$  pixels can be approximately reconstructed by roughly 100 to 500 base images.

The idea that natural images may be represented with sparse coding reflects the nature of the point cloud formed by the population of natural images, which has many low-dimensional spikes, as illustrated by Fig. 2.6. Indeed, as we will see in the following, high kurtosis motivates sparse representations.

The phenomenon described that involved 1D marginal histogram spikes occurring in the 3D space of the data distribution motivates the idea of sparse coding. Consider each low-dimensional spike to be a street and each location on that spike to be an address. With a street name and address, one can use the low-dimensional spikes to “travel” to anywhere in the data distribution. The street is the filter  $F$ , and the address

is the filter response.

Now consider an image reconstruction

$$\mathbf{I} = \sum_{i=1}^k a_i F_i + \epsilon$$

in which  $\mathbf{I}$  is an image,  $k$  is the number of active neurons,  $\epsilon$  is the residual,  $a_i$  is the filter response,  $F_i$  is the filter, and  $N^2$  is the size of the image. Compact coding states that a vector  $\mathbf{I} \in R^2$ , projected to a low-dimensional space, can be encoded into  $K$  neurons in a non-orthogonal manner with few principal components, in which  $K \ll N^2$ .

A sparse representation is an opposite. For sparse coding,  $M$  neurons are used to encode  $N^2$  representations, in which  $N^2 \ll M$ . For sparse coding, an overcomplete basis is used, primarily composed of zeros for a total of  $M$  encodings, many of which are redundant. How does one come up with the important parts of an image to represent, i.e., the  $M$  encodings? The low-dimensional spikes for Fig. 2.6 are used. Recall that each filter response reveals a statistical property of the distribution, so together they may form a base dictionary for image reconstruction. Analogously, when one asks for directions to reach a town, usually the response will include street names and other low-dimensional descriptions or features, just as is done in sparse coding. A key point to remember, then, is that the point cloud formed by natural images contains many low-dimensional spikes, reflective of the high kurtosis of natural images, which may be used as guides, or viable directions, to traverse and better learn or represent the image space.

## 2.5 Scale Invariance

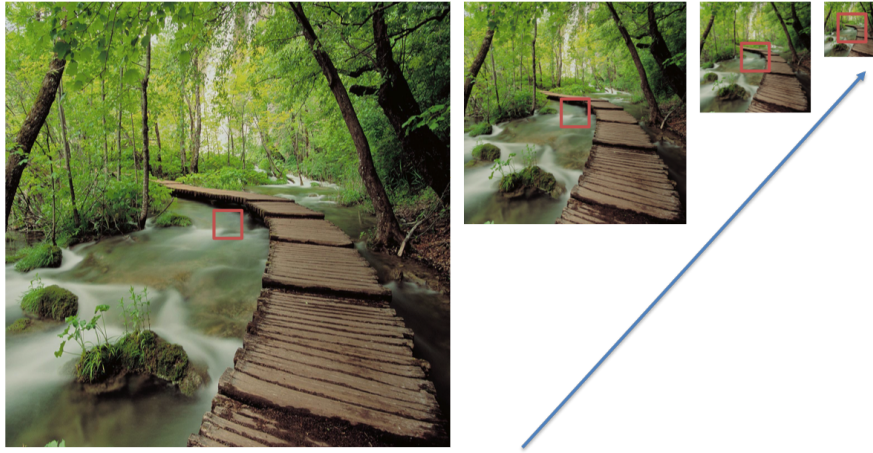


Figure 2.8: A natural image scaled through 1x, 2x, 4x, and 8x downsampling can be observed. Although the fixed-size filter kernel (shown in red) captures more information for each downsampled scale, the image statistics remain invariant.

In addition to the heavy-tailed histograms of Gabor filter responses, another interesting observation reported by Ruderman (94) [192] and Zhu and Mumford (97) [261] is that histograms of gradient-filtered images are consistent over a range of scales (see Fig. 10.4). This is the fourth statistical observation for natural images: scale invariance. Specifically, for a natural image  $\mathbf{I}$ , a pyramid with a number of  $n$  scales can be built, such that  $\mathbf{I} = \mathbf{I}^{(0)}, \mathbf{I}^{(1)}, \dots, \mathbf{I}^{(n)}$ , and  $\mathbf{I}^{(s+1)}$  is obtained by averaging every block of  $2 \times 2$  pixels

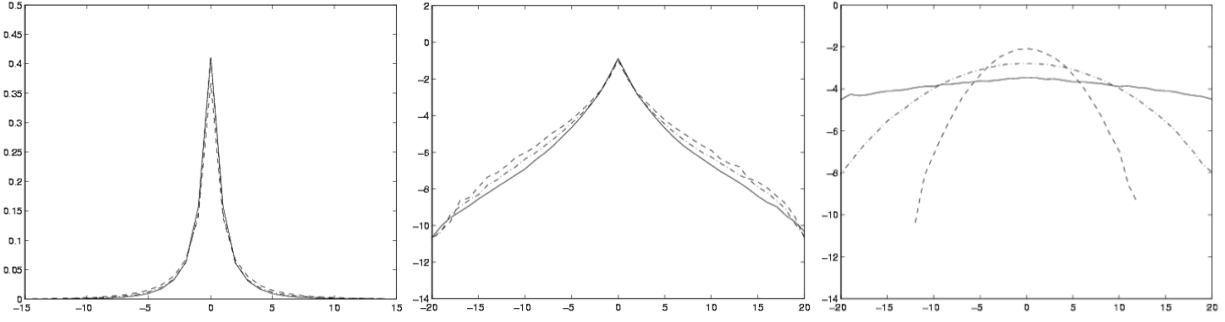


Figure 2.9: First two plots: histograms of gradient-filtered natural image from Fig. 2.8 at three different scales. Last plot: log of the histograms of gradient-filtered Gaussian noise image at the same three scales. From this, it can be observed that high kurtosis is a property of natural images and not, e.g., Gaussian noise, and the level of kurtosis persists over different scales.

in  $\mathbf{I}^{(s)}$ . The histograms of gradients  $\nabla_1 \mathbf{I}^{(s)}(x)$  for natural images (first two plots) and for Gaussian noise (last plot) are shown in Fig. 10.4, for three scales  $s = 1, 2, 3$  shown in Fig. 2.8. Note that, for natural images, despite scaling down the image, roughly the same amount of kurtosis is observed in the histograms for three different scales.

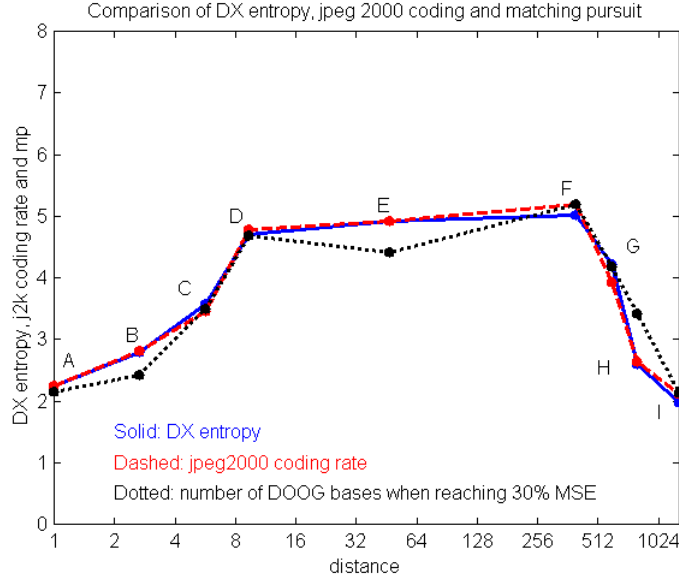


Figure 2.10: Reprinted with permission from [235]. Scale invariance drops off after a certain point.

To better illustrate the scale invariance property of natural images, here a toy example is presented: a 2D invariant world consisting of only 1D line segments. Fig. 2.11 (a) shows an image of the simulated world. In the image, each line segment is determined uniquely by its center location  $(x_i, y_i)$ , orientation  $\theta_i$ , and length  $r_i$ . The lines are independently distributed in terms of geometric features. The center of a line is selected from the image plane uniformly. Its orientation, measured as the angle formed with the horizontal line, is also uniformly distributed over  $[0, \pi]$ . The length of a line follows a cubic probability density function, i.e.,  $p(r) \propto \frac{1}{r^3}$ . In reality,  $r_i$  can be sampled from  $p(r)$  using inverse transform sampling. While the lines are

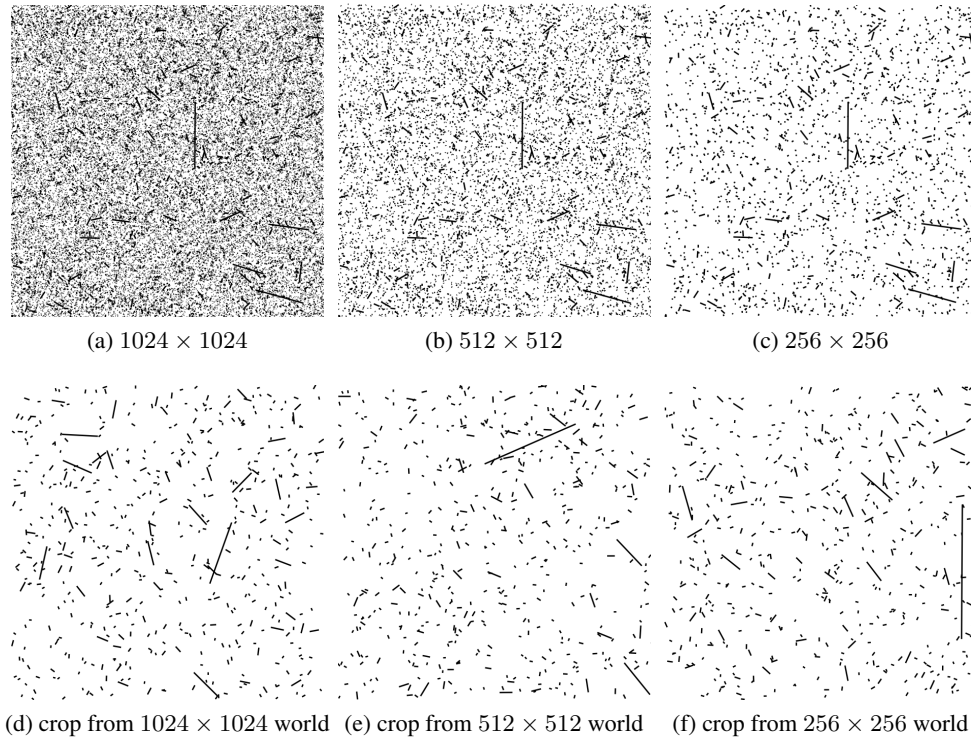


Figure 2.11: The top three figures represent the 2D toy world with different pixel sizes; the bottom ones are  $128 \times 128$  pixel crops from the corresponding figures.

i.i.d. in geometric features, the overall density of the segments is controlled by a Poisson distribution. That is, in each unit area, the number of line segments has a constant mean. The toy world can be constructed by the above rules, and it can be observed that it has the scale-invariant property similar to that of natural images. To be more specific, Fig. 2.11 (a) is of size  $1024 \times 1024$  pixels. Fig. 2.11 (b) and (c) are obtained by down-sampling the original image to  $512 \times 512$  pixels and  $256 \times 256$  pixels, respectively. Notice that in down-sampling, the long lines are truncated and the lines shorter than a pixel are discarded. Then,  $128 \times 128$  pixel patches can be cropped from these three images. Indeed, the three crops are identical to each other and hard to tell apart. This shows that the 2D toy world is scale-invariant in the sense that the image features are identical across scales. In fact, parallels can be drawn between the geometric features of the line segments and the gradient histograms of natural images. These statistics remain the same no matter which scale the images are viewed.



# 3

## Textures

Texture is an important characteristic of the appearance of objects in natural scenes and is a crucial cue in visual perception. It plays an important role in computer vision, graphics, and image encoding. Understanding texture is an essential part of understanding human vision.

Texture analysis and synthesis has been an active research area, and a large number of methods have been proposed, with different objectives or assumptions about the underlying texture formation processes. In computer vision and psychology, instead of modeling specific texture formation processes, the goal is to search for a general model which should be able to describe a wide variety of textures in a common framework, and which should also be consistent with the psychophysical and physiological understanding of human texture perception.

### 3.1 The Julesz Quest

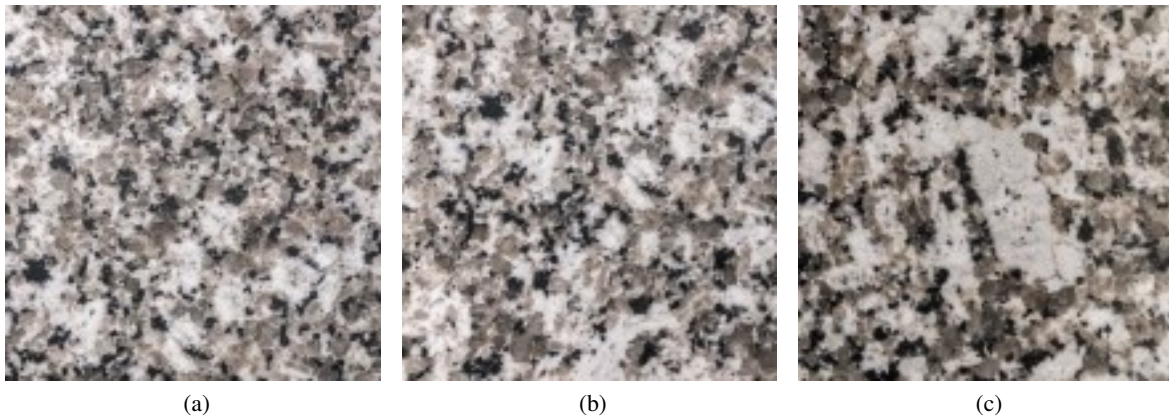


Figure 3.1: (a) presents a section of the marble you observe. (b) presents another section of the same marble piece. (c) is a defective area of the marble.

Imagine a scenario in which you walk into a store to purchase marble tiles for your new home. You search for the most consistent pattern, so your new floor looks uniform and the pieces are indistinguishable from one another. Suddenly, you notice a piece of marble, as shown in Fig. 3.1a and Fig. 3.1b, whose patterns you appreciate. However, as you continue inspecting this piece of marble, you suddenly catch



attention of an area such as depicted in Fig. 3.1c. You ask yourself the question, “Is the texture in this marble patch consistent with the rest of the marble?” “What features of a texture make it distinguishable from another texture?” These questions may seem ostensibly simple to answer, but it is fundamentally difficult to exactly define the distinguishing features that set textures apart from one another.

Differentiating differences in texture patterns seems an intuitive and easy task for humans, but why are we able to differentiate the textures so easily? “The defects in the texture,” one may answer, but how are the raw visual signals converted so that human brains can easily distinguish the defective patterns? How are texture patterns represented in human brains in general? This is a question that has troubled psychologists for many years until Julesz initiated formal research on texture modeling.

In his seminal paper in 1962 [113], Julesz conducted research on textures and asked the following fundamental question: *What features and statistics are characteristic of a texture pattern, so that texture pairs that share the same features and statistics cannot be told apart by pre-attentive human visual perception?*

As illustrated by Fig. 3.2, the texture pattern formed by the + signs pops out from the surrounding pattern at first glance (so-called pre-attentive vision) in the left panel, but it merges with the surrounding pattern in the right panel. We have to examine the right panel attentively in order to tell the two texture patterns apart.

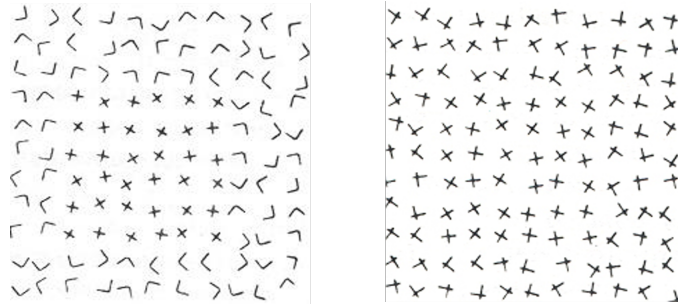


Figure 3.2: Julesz texture pairs: the texture pattern formed by the + signs pops out from the surrounding pattern in the left panel, but it merges with the surrounding pattern in the right panel.

The first general texture model was proposed by Julesz in the 1960s. Julesz suggested that texture perception might be explained by extracting the so-called ‘ $k$ -th order’ statistics, i.e., the co-occurrence statistics for intensities at  $k$ -tuples of pixels [113]. A key drawback of this model is that, on one hand, the amount of data contained in the  $k$ -th order statistics is big and thus very hard to handle when  $k > 2$ . On the other hand, psychophysical experiments show that the human visual system does extract at least some statistics of order higher than 2 [39].

In mathematical terms, a set of all texture images with the same features and statistics can be written as

$$\Omega_{\mathbf{I}} = \{\mathbf{I} : \mathbf{H}_i(\mathbf{I}) = \mathbf{h}_i, i = 1, \dots, K\} \quad (3.1)$$

where  $\mathbf{I}$  is a texture, each  $\mathbf{H}_i$  is a chosen feature/statistics, and  $K$  is the number of features chosen to be extracted such that texture patterns become indistinguishable when all features are extracted and matched. The search for the  $\mathbf{H}_i$  has gone a long way beyond Julesz’s statistical conjecture. Methods employed include co-occurrence matrices, run-length statistics, sizes and orientations of various textons, cliques in Markov Random Fields, and dozens of other measures. All these features have rather limited expressive power.

Here, it is worth noting that the modeling of texture concerns not only the data and distributions in the image space but also depends on human perception influenced by task, or purpose. In Julesz’s quest, the criterion for judging whether two texture images belong to the same category depends on the human visual



system, which is trained for various tasks. Julesz carefully reduces task-dependence by testing in a pre-attentive stage with no specific purpose; nevertheless, the human vision system has been trained ahead of the experiments. Later experiments show that human vision can learn and adapt to tell apart texture images after long training. This task-dependence sets apart models in vision from those in physics.

In the subsequent chapters, let us illuminate the quest for features and statistics describing texture patterns.

## 3.2 MRF & Clique-Based Gibbs Models

One approach for pursuing texture features is statistical modeling, which characterizes texture images as arising from probability distributions on random fields [29]. These modeling approaches involve only a small number of parameters and thus provide concise representation for textures. More importantly, texture analysis can be posed as a well-defined statistical inference problem. These statistical approaches enable us to not only infer the parameters of underlying probability models of texture images but also to synthesize texture images by sampling from these models. Checking whether the synthesized images have similar visual appearances to the textures being modeled provides a rigorous way to test the model.

An issue, however, is that many of these statistical models, such as Markov random fields and clique-based Gibbs models, are too simple and thus suffer from a lack of expressive power to capture the fidelity of natural images.

### 3.2.1 Markov Random Fields (MRF)

Markov Random Fields (MRF) models were popularized by Besag in 1973 [17] for modeling spatial interactions on lattice systems and were used by Cross and Jain in 1983 [29] for texture modeling. An important characteristic of MRF modeling is that global patterns are formed via stochastic propagation of local interactions, which is particularly appropriate for modeling textures as they are characterized by global but not predictable repetitions of similar local structures.

Concretely, a Markov random field is a probability distribution over random variables  $X_1, \dots, X_n$  defined by an undirected graph  $G = (V, E)$ , where the nodes in  $V$  correspond to the variables  $X_i$ . Before illustrating the special properties of MRF models, we first introduce some notations to formalize graphs. In an undirected graph  $G$ , two nodes  $s, t$  are called neighbors if there is an edge between them. The neighborhood of a site  $s$  is defined as the set of all its neighbors, i.e.,  $\mathcal{N}_s = \{t : (s, t) \in E, t \in V\}$ . Moreover, the neighborhood system of  $G$  is the set of all neighborhoods in the graph. We denote it as  $\mathcal{N} = \{\mathcal{N}_s : s \in V\}$ . For any patch  $A$  in graph  $G$ , define  $X_A$  as a window of observation. The boundary of patch  $A$  is thus  $\mathcal{N}_A = \{t : (s, t) \in E, s \in A, t \notin A\}$ . Also, we call  $C \subset V$  a clique of  $G$  if for any two sites  $s, t \in C$ , we have  $(s, t) \in E$ . The depiction on the right-hand side of Fig. 3.3 shows a simple general MRF, in which, for example, the neighbors of node  $B$  are  $C, D$  and  $E$ , and nodes  $B, D$ , and  $E$  form a three-node clique. A distinct feature of MRF models is that the random variables defined on the nodes satisfy the local Markov properties. That is, each variable  $X_i$  is conditionally independent of all other variables given its neighbors. Therefore, MRF can be used to model a set of local characteristics.

Based on the qualities of MRF models, we can consider a texture as a realization from a random field  $\mathbf{I}$  defined over a spatial configuration  $\mathcal{D}$ . For example,  $\mathcal{D}$  can be an array or a lattice. We denote  $\mathbf{I}_s$  as the random variable at a location  $s \in \mathcal{D}$ , and let  $\mathcal{N} = \{\mathcal{N}_s, s \in \mathcal{D}\}$  be the neighborhood system of  $\mathcal{D}$  satisfying: 1)  $s \notin \mathcal{N}_s$  and 2)  $s \in \mathcal{N}_t \iff t \in \mathcal{N}_s$ . The pixels in  $\mathcal{N}_s$  are the neighbors of  $s$ . A subset  $C$  of  $\mathcal{D}$  is a clique if every pair of distinct pixels in  $C$  are neighbors of each other;  $\mathcal{C}$  denotes the set of all cliques.

Formally, the following gives a definition of MRF distributions on images.

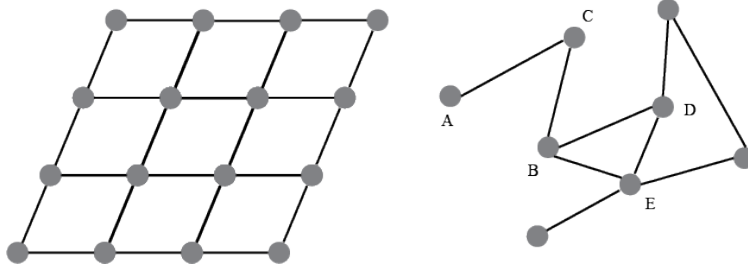


Figure 3.3: On the left is an example lattice structure of a MRF. On the right is a toy example of a general MRF in which  $A$  and  $C$  are neighbors of each other, thus forming a two-node clique, and  $B, D, E$  form a three-node clique.

**Definition.**  $p(\mathbf{I})$  is an MRF distribution with respect to  $\mathcal{N}$  if  $p(\mathbf{I}_s | \mathbf{I}_{-s}) = p(\mathbf{I}_s | \mathbf{I}_{\mathcal{N}_s})$ , in which  $\mathbf{I}_{-s}$  denotes the values of all pixels other than  $s$ , and for  $A \subset \mathcal{D}$ ,  $\mathbf{I}_A$  denotes the values of all pixels in  $A$ .

In addition, an MRF distribution is closely related to a Gibbs distribution, which is defined below. In statistical mechanics, a Gibbs distribution gives the probability that a system will be in a certain state as a function of the state's local properties, such as the energy and the temperature of the system. Consequently, the Gibbs model can be viewed as a set of potentials defined on cliques.

**Definition.**  $p(\mathbf{I})$  is a Gibbs distribution with respect to  $\mathcal{N}$  if

$$p(\mathbf{I}) = \frac{1}{Z} \exp\left\{-\sum_{C \in \mathcal{C}} \lambda_C(\mathbf{I}(C))\right\}, \quad (3.2)$$

where  $Z$  is the normalizing constant (or partition function), and  $\lambda_C(\cdot)$  is a function of intensities of pixels in clique  $C$  (called potential of  $C$ ). Some constraints can be imposed on  $\lambda_C$  for them to be uniquely determined.

In fact, an MRF distribution is equivalent to a Gibbs distribution. The Hammersley-Clifford theorem establishes their equivalence [17]:

**Theorem 1.** For a given  $\mathcal{N}$ ,  $p(\mathbf{I})$  is an MRF distribution  $\iff p(\mathbf{I})$  is a Gibbs distribution.

This equivalence provides a general method for specifying an MRF on  $\mathcal{D}$ , i.e., first choose an  $\mathcal{N}$ , and then specify  $\lambda_C$ . The MRF is *stationary* if for every  $C \in \mathcal{C}$ ,  $\lambda_C$  depends only on the relative positions of its pixels. This is often pre-assumed in texture modeling.

Here the  $\lambda_C(\cdot)$  function, which extracts statistics from an input image, can be seen as the feature representation  $\mathbf{H}_i$  of the image as introduced in Section 3.1.

Often enough, a texture is considered as an MRF on a lattice system with each pixel represented by a node, as shown on the left of Fig. 3.3. The neighboring pixels form a clique and pixels farther away have less effect on the pixel in question. This paired system leads us to consider auto-models (Besag 1973), MRF models with pair potentials.

### 3.2.2 Ising & Potts Model

Two important instantiations of MRF models with pair potentials have emerged throughout history, the Ising and the Potts model.

In general, auto-models with pair potentials have characteristics of  $\lambda_C \equiv 0$  if  $|C| > 2$  where  $p(\mathbf{I})$  has the following form:

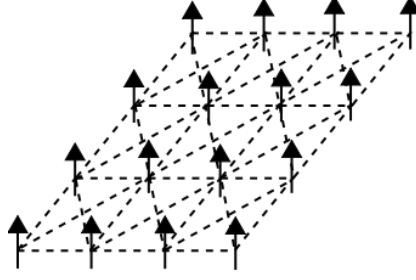


Figure 3.4: Graphical view of Ising Model with all positive spin.

$$p(\mathbf{I}) = \frac{1}{Z} \exp\left\{\sum_s \alpha_s \mathbf{I}_s + \sum_{t,s} \beta_{t-s} \mathbf{I}_t \mathbf{I}_s\right\}, \quad (3.3)$$

in which  $\beta_{-t} = \beta_t$  and  $\beta_{t-s} \equiv 0$  unless  $t$  and  $s$  are neighbors. An MRF model with pair potentials, as defined above, is commonly specified through conditional distributions:

$$p(\mathbf{I}_s \mid \mathbf{I}_{-s}) \propto \exp\left\{\sum_s \alpha_s \mathbf{I}_s + \sum_t \beta_{s-t} \mathbf{I}_t \mathbf{I}_s\right\},$$

in which the neighborhood is usually of order less than or equal to three pixels.

One of the classic MRF models with pair potentials is the Ising model. The Ising model was first proposed by Ernst Ising to study ferromagnetism. Similar to texture models, the Ising Model also considers lattice systems, in which each node  $\mathbf{I}_s \in \{+1, -1\}$ . To attach physical meaning to this construction, each node, or site, can be viewed as an electron having a particular “spin.” With  $\mathbf{I}_s = -1$ , electron  $s$  points down, and with  $\mathbf{I}_s = +1$ , the electron points up. An Ising Model with all positive spins is shown in Fig. 3.4.

How do the electrons in the lattice interact with each other? What is the energy of the system? To answer these questions, the Ising Model considers two types of interactions that affect the energy of the system: the *external field* and *interaction* between neighboring electrons. Together, they form an energy function called a *Hamiltonian*, written as

$$H(\mathbf{I}) = - \sum_{\langle s,t \rangle} \beta \mathbf{I}_s \mathbf{I}_t - \sum_t \alpha \mathbf{I}_t \quad (3.4)$$

in which  $\beta$  represents the strength of magnetization and dictates electron interactions, and  $\alpha$  represents strength of external field on each electron.

The entire configuration of the lattice system is, similar to a general MRF, given by

$$p(\mathbf{I}) = \frac{1}{Z} \exp\left\{-\frac{H(\mathbf{I})}{kT}\right\} \quad (3.5)$$

in which  $k$  is the Boltzmann constant and  $T$  is the temperature. The higher the temperature, the more random the particles become, and the more uniform the probability distribution becomes. An example sampling of the Ising Model using different  $\beta$  values is shown in Fig. 3.5.

The Potts Model is a generalization of the Ising Model on a lattice system. Instead of binary states, each particle has  $n$  spin angles, i.e.,  $\mathbf{I}_s \in \{\theta_0, \theta_1, \dots, \theta_{n-1}\}$ . Therefore, the energy Hamiltonian is given by



Figure 3.5: Example sampling of Ising Model with different mean and variance. From left to right,  $\beta$  is 0.35, 0.40, and 0.43, respectively.

$$H(\mathbf{I}) = - \sum_{\langle s,t \rangle} \beta \delta(\mathbf{I}_s, \mathbf{I}_t) - \sum_t \alpha \mathbf{I}_t \quad (3.6)$$

in which  $\delta(\cdot, \cdot)$  is the Kronecker delta function. Its distribution is in the same form as that of the Ising Model.

The advantage of pair-potential models such as the Ising and Potts models is that the parameters in the models can be easily inferred by auto-regression. However, these models are severely limited in the following two ways: 1) the cliques are too small to capture features of texture, and 2) the statistics on the cliques specify only first-order and second-order moments, i.e., mean and covariance, respectively. Yet many textures have local structures much larger than three or four pixels, and the covariance information, or equivalently the spectrum, cannot adequately characterize textures, as suggested by the existence of distinguishable texture pairs with identical second-order or even third-order moments, as well as indistinguishable texture pairs with different second-order moments [39]. Moreover, many textures are strongly non-Gaussian, regardless of neighborhood size.

The underlying reason for these limitations is that Equation 3.2 involves too many parameters if we increase the neighborhood size or the order of the statistics, even for the simplest auto-models. This suggests that we need carefully designed functional forms for  $\lambda_C(\cdot)$  to efficiently characterize local interactions as well as the statistics on local interactions.

### 3.2.3 Gaussian Markov Random Fields (GMRF)

An early statistical model for texture patterns was the Gaussian Markov Random Field (GMRF). This family of models satisfies the properties of an MRF model with the additional restriction that the joint distribution of all nodes is multivariate normal. Important statistical properties of natural images can be observed in the GMRF model, suggesting that certain basic features of natural images are represented by this well-defined parametric family. The idea of matching fundamental features of observed images to synthesized new images with the same appearance provides inspiration for more complex models such as the FRAME model (see Section 3.4).

Formally, a GMRF is a graph  $\mathcal{G} = (\mathcal{V}, \mathcal{E})$  where each node corresponds to a single dimension of a multivariate Gaussian  $x \sim \mathcal{N}(\mu, Q^{-1})$  with a non-singular precision matrix  $Q$ . Edges of  $\mathcal{G}$  are determined by the relation

$$Q_{k,\ell} \neq 0 \Leftrightarrow \{k, \ell\} \in \mathcal{E}.$$

Conditional independence satisfies  $x_k \perp x_\ell \mid x_{-k\ell} \Leftrightarrow Q_{k,\ell} = 0$ , as can be seen from direct inspection of the multivariate Gaussian density. Therefore,  $x_k \mid x_{-k} \sim x_k \mid \mathcal{N}(x_k)$ , in which  $\mathcal{N}(x_k) = \{x_\ell : Q_{k,\ell} \neq 0, k \neq \ell\}$ .

$\ell\}$  is the neighborhood of  $x_k$ , i.e., the distribution of a single node  $x_k$  given the rest of the nodes depends only on a subset of nodes which are connected to  $x_k$  in the GMRF graph. Note that the edges often represent nodes that are spatially related, such as nearby pixels within an image.

Consider an  $N \times M$  image  $\mathbf{I} \sim \mathcal{N}(0, \beta^{-1}Q^{-1})$ , in which  $\mathbf{I}(x, y)$  denotes pixel at location  $(x, y)$  (note that this is different from the previous graph structure notation, in which  $\mathbf{I}_s$  denotes node  $s$  in  $\mathbf{I}$ ),  $\beta > 0$  is the coupling strength between pixels, and

$$Q_{(x_1, y_1), (x_2, y_2)} = \begin{cases} 1 & (x_1, y_1) = (x_2, y_2) \\ -\frac{1}{4} & (x_2, y_2) \in \mathcal{N}(\mathbf{I}(x_1, y_1)) \\ 0 & \text{else} \end{cases}$$

with neighborhood structure  $\mathcal{N}(\mathbf{I}(x, y)) = \{\mathbf{I}(x+1, y), \mathbf{I}(x-1, y), \mathbf{I}(x, y+1), \mathbf{I}(x, y-1)\}$ . Suppose that the torus boundary condition is used, so that  $\mathbf{I}(x+N, y) = \mathbf{I}(x, y)$  and  $\mathbf{I}(x, y+M) = \mathbf{I}(x, y)$ . This model has the density

$$p(\mathbf{I}) = \frac{1}{Z} \exp \left\{ -\beta \sum_x \sum_y (\mathbf{I}(x+1, y) - \mathbf{I}(x, y))^2 + (\mathbf{I}(x, y+1) - \mathbf{I}(x, y))^2 \right\}, \quad (3.7)$$

and the conditional expectation for each pixel

$$\mathbb{E}[\mathbf{I}(x, y) | \mathcal{N}(\mathbf{I}(x, y))] = \frac{1}{4} (\mathbf{I}(x+1, y) + \mathbf{I}(x-1, y) + \mathbf{I}(x, y+1) + \mathbf{I}(x, y-1))$$

is the average of the neighboring pixels in the image. In the limiting case of infinitesimally small pixels, the discrete GMRF model converges to the continuous density

$$p(\mathbf{I}) = \frac{1}{Z} \exp \left\{ -\beta \int_x \int_y (\nabla_x \mathbf{I}(x, y))^2 + (\nabla_y \mathbf{I}(x, y))^2 dy dx \right\}. \quad (3.8)$$

Several important observations follow from analysis of the continuous analogue of the discrete GMRF density. First, the power law phenomenon observed in natural images can be explicitly derived for the density in Equation 3.8. Recall that the Fourier transform  $\hat{\mathbf{I}}(\xi, \eta)$  of an image  $\mathbf{I}(x, y)$  is given by

$$F(\mathbf{I}) = \hat{\mathbf{I}}(\xi, \eta) = \int_x \int_y \mathbf{I}(x, y) e^{-i2\pi(x\xi + y\eta)} dy dx.$$

Algebraic manipulation shows that  $F(\nabla_x \mathbf{I}) = 2\pi i \xi \hat{\mathbf{I}}$  and  $F(\nabla_y \mathbf{I}) = 2\pi i \eta \hat{\mathbf{I}}$ . The well-known Plancherel Theorem states that a function  $g$  and its Fourier transform  $G = F(g)$  satisfy the relation

$$\int |g(t)|^2 dt = \int |G(\eta)|^2 d\eta,$$

meaning that the  $L_2$  functional norm is preserved by the Fourier transform. Bringing all this together shows

$$\beta \int_x \int_y (\nabla_x \mathbf{I}(x, y))^2 + (\nabla_y \mathbf{I}(x, y))^2 dy dx = 4\pi^2 \beta \int_\xi \int_\eta (\xi^2 + \eta^2) |\hat{\mathbf{I}}(\xi, \eta)|^2 d\eta d\xi$$

so that the potential function in Equation 3.8 can be rewritten in terms of  $\hat{\mathbf{I}}$ . Moreover, the separable form of the right-hand side of the above equation shows

$$p(\hat{\mathbf{I}}(\xi, \eta)) \propto \exp \left\{ -4\pi^2 \beta (\xi^2 + \eta^2) |\hat{\mathbf{I}}(\xi, \eta)|^2 \right\}.$$

Hence,  $\hat{\mathbf{I}}(\xi, \eta)$  is independent of other states of  $\hat{\mathbf{I}}$ , and  $\hat{\mathbf{I}}(\xi, \eta)$  is a Gaussian with parameters

$$\mathbb{E}[\hat{\mathbf{I}}(\xi, \eta)] = 0 \quad \text{and} \quad \text{Var}[\hat{\mathbf{I}}(\xi, \eta)] = \frac{1}{8\pi^2\beta(\xi^2 + \eta^2)}.$$

Therefore,  $\mathbb{E}[|\hat{\mathbf{I}}(f)|^2]^{1/2} \propto \frac{1}{f}$  so that the GMRF model  $p(\mathbf{I})$  satisfies the scale invariance observed in natural images, as discussed in Section 2.3.

A second important property of the GMRF model is its connection to the heat equation. Considering again the potential function of a GMRF model

$$H(\mathbf{I}(x, y)) = \beta \int_x \int_y (\nabla_x \mathbf{I}(x, y))^2 + (\nabla_y \mathbf{I}(x, y))^2 dy dx,$$

we will show that generating an image according to the dynamics

$$\frac{d\mathbf{I}(x, y, t)}{dt} = -\frac{\delta H(\mathbf{I}(x, y, t))}{\delta \mathbf{I}}, \quad \forall x, y$$

is equivalent to the heat-diffusion equation

$$\frac{d\mathbf{I}(x, y, t)}{dt} = \Delta \mathbf{I}(x, y),$$

in which  $\Delta = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$  is the Laplacian operator.

Using an Euler-Lagrangian equation for two variables,

$$\frac{\delta J}{\delta f} = \frac{\partial L}{\partial f} - \frac{d}{dx} \left( \frac{\partial L}{\partial f_x} \right) - \frac{d}{dy} \left( \frac{\partial L}{\partial f_y} \right),$$

in which  $J(f) = \int_y \int_x L(x, y, f, f_x, f_y) dx dy$  and  $f$  is a function of  $x$  and  $y$ .

Setting  $f = \mathbf{I}$ ,  $L = (\nabla_x \mathbf{I})^2 + (\nabla_y \mathbf{I})^2$ , and  $J = H$ , we obtain

$$\begin{aligned} \frac{d\mathbf{I}(x, y, t)}{dt} &= -\frac{\delta H(\mathbf{I}(x, y, t))}{\delta \mathbf{I}} \\ &= -\left[ \frac{\partial[(\nabla_x \mathbf{I})^2 + (\nabla_y \mathbf{I})^2]}{\partial \mathbf{I}} - \frac{d}{dx} \left[ \frac{\partial[(\nabla_x \mathbf{I})^2 + (\nabla_y \mathbf{I})^2]}{\partial(\nabla_x \mathbf{I})} \right] - \frac{d}{dy} \left[ \frac{\partial[(\nabla_x \mathbf{I})^2 + (\nabla_y \mathbf{I})^2]}{\partial(\nabla_y \mathbf{I})} \right] \right] \\ &= 0 + \frac{d}{dx} [2(\nabla_x \mathbf{I})] + \frac{d}{dy} [2(\nabla_y \mathbf{I})] \\ &= 2\left(\frac{\partial^2 \mathbf{I}}{\partial x^2} + \frac{\partial^2 \mathbf{I}}{\partial y^2}\right) \\ &= 2\Delta \mathbf{I}. \end{aligned}$$

Ignoring the constant factor, we have shown that in fact the learning dynamics of an image modeled by a GMRF potential function is equivalent to the dynamics modeled by the heat-diffusion equation.

Despite interesting connections to statistical properties of natural images and the heat-diffusion equation, the GMRF model is still quite restricted. Interesting local structures exist but the joint pixel density is a unimodal Gaussian. The GMRF model can capture aspects of the local regularity found in natural images but nothing else. Therefore, the GMRF model from Equation 3.7 is not capable of serving as a probability density for even simple image patterns. The GMRF model can represent certain properties of an image (specifically, the tendency of nearby pixels to have similar intensity), but it cannot account for the formation of visual patterns. In the following sections, we will see how the GMRF potential from Equation 3.7 can be adapted as one of many filters (in particular, a gradient filter) whose joint features can be used to synthesize realistic images.

### 3.2.4 Advanced Models: Hierarchical MRF and Mumford-Shah Model

In reality, we often encounter images with multiple types of textures, each occupying a set of regions in the image domain. In these situations, we need more powerful models to deal with the interaction of distinct texture regions.

When several texture patterns are present in an image, the boundaries of different regions need to be specified. Often, the simple MRF models, including the GMRF model, are not expressive enough in encoding this kind of knowledge in images. For example, we might expect long, straight edges to be penalized in some images, but the pixel intensities alone do not accurately reflect the existence of edge elements. Geman and Geman [70] introduce the hierarchical MRF model for the maximum a posteriori (MAP) image restoration problem and describe the images by both pixel intensities and edge continuity. Specifically, an image  $\mathbf{I} = (\mathbf{F}, \mathbf{L})$  is modeled as an MRF of two processes. The intensity process  $\mathbf{F}$  is a simple MRF of observable pixel intensities as discussed in Section 3.2.3). The line process  $\mathbf{L}$  is another MRF of unobservable edge elements. We define a line site  $d$  as placed midway between each vertical or horizontal pair of pixels. The set  $\mathcal{D}$  of all line sites is thus all possible locations of the edge elements. While  $\mathbf{F}(x, y)$  measures the intensity at a pixel,  $\mathbf{L}(x, y)$  represents a lack, or presence (and orientation) of an edge at this specific site. The line process  $\mathbf{L}$  can affect a pixel's neighborhood. For instance, if an edge is present at a line site, the potential over the pair-clique consisting of the pixels separated by this line site is zero. Therefore, these two pixels will not influence the other's intensity, and the bonding between them is broken. The Gibbs distribution used to define the hierarchical MRF is

$$p(\mathbf{F} = f, \mathbf{L} = l) = \frac{1}{Z} \exp\left\{-\sum_{C \in \mathcal{C}} \lambda_C(f, l)\right\}. \quad (3.9)$$

With the hierarchical MRF, we can model more complicated image structures and define new types of tasks in computer vision, as it allows us to analyze the discontinuity in images from a new perspective.

Another technique that facilitates the modeling of multiple textures is the Mumford-Shah model. This model aims to segment an image into a few simple regions, keeping the color of each region as smooth as possible. It utilizes the Mumford-Shah energy functional to complete three tasks simultaneously. First, the functional measures how good the model approximates the observed image. Second, it asks that the model varies slowly except at boundaries. Third, it requires the set of contours to be as short, and hence as simple and straight, as possible.

### 3.2.5 Selecting Filters and Learning Potential Functions

The models mentioned thus far have equipped us with the tools needed to describe and generate a variety of textures, as long as we select appropriate image filters and learn optimal potential functions and parameters for the learned probability distributions. The questions then become, given one or a set of observed texture images, how do we select filters that will most completely capture the image statistics? And how do we learn the optimal potential functions and parameters? These questions will be answered as we expound the FRAME model in Section 3.4. First though, it will be helpful to review some of the more well-known filters in vision.

## 3.3 Filters for Early Vision

Due to the limitations of clique-based models, researchers have also explored feature extraction from the perspective of image filtering. In the various stages along the visual pathway, from the retina, to V1, to

the extra-striate cortex, cells with increasing sophistication and abstraction have been discovered: center-surround isotropic retinal ganglion cells, frequency and orientation selective simple cells, and complex cells that perform non-linear operations. Here we focus on filtering theory inspired by the multi-channel filtering mechanism discovered and generally accepted in neurophysiology, which proposes that the visual system decomposes the retinal image into a set of sub-bands. These sub-bands are computed by convolving the image with a bank of linear filters, followed by some non-linear procedures [16]. Considering again the definition  $\Omega(\mathbf{h}) = \{\mathbf{I} : \mathbf{H}_i(\mathbf{I}) = \mathbf{h}_i, i = 1, \dots, K\}$ , it is now natural, after convolving the image with a bank of linear filters, to use a marginal statistic (e.g., a histogram) to represent a feature  $\mathbf{H}_i$  of an image.

Filtering is a process that changes pixel values of a given image, in an effort to extract valuable information such as clusters and edges. Fig. 3.6 provides an example of typical image filtering.

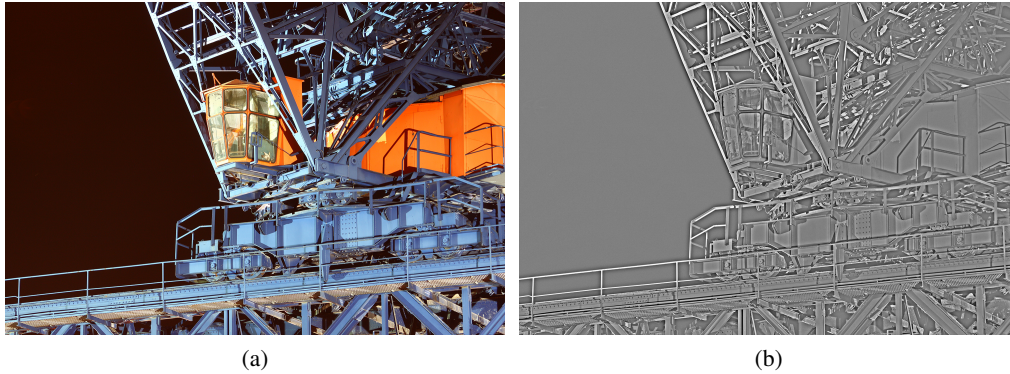


Figure 3.6: (a) presents an image and (b) presents a filtered version of the image.

An integral part of image processing, filters provide biologically plausible ways of extracting visual information from raw input signals. In the coming section, we will explore different types of filters that extract different feature information from images.

### 3.3.1 Correlation & Convolution

Before we introduce specific types of filters, it is important to understand how filters are applied to images. One method of linear filtering is correlation, which uses filters to obtain a weighted combination of all pixels in a small neighborhood. Suppose an image  $\mathbf{I}$  is defined on an  $N \times N$  lattice  $D$ , and  $\mathbf{I}(x, y) \in \mathcal{L}$  is the intensity value of an image  $\mathbf{I}$  at location  $(x, y)$  and  $\mathcal{L}$  is an interval on the real line or a set of integers representing pixel intensity. Suppose also that filter  $F$  is defined on an  $M \times M$  lattice much smaller than the lattice of  $\mathbf{I}$ . Correlation is computed as

$$\mathbf{I}(x, y) \otimes F = \sum_{(k, l)} \mathbf{I}(x + k, y + l) F(k, l) \quad (3.10)$$

in which  $\otimes$  denotes correlation and each  $F(k, l)$  is a filter coefficient. To maintain the image size, we pad the boundary of the original image with enough zeros so that the filtered output image is the same size.

Correlation measures similarity between a filtered region and the filter itself, but to extract meaningful information, a more commonly used method is convolution. Different from correlation, convolution is defined as

$$\mathbf{I}(x, y) * F = \sum_{(k, l)} \mathbf{I}(x - k, y - l) F(k, l) \quad (3.11)$$



where  $*$  denotes the convolution operation. It appears that convolution is correlation with both axes flipped. The motivation for using convolution instead of correlation is that it is associative, in addition to being commutative and distributive. This means that if an image needs to be convolved with multiple filters, the filters can be convolved with each other first before being applied to the image. A detailed proof of the mathematical properties of convolution is not of our interest for this book, but to intuit why convolution is associative, notice that the Fourier transform of convolution is a product in the frequency domain, which is clearly associative.

Knowing some basic operations for image processing, let us introduce some classical filters.

### 3.3.2 Edge Detection Filters

An edge in an image consists of a sudden change of pixel intensity, and to detect them, a natural decision is to detect the change of pixel values throughout the image. If an image exists on a continuous domain, we can represent change in pixel intensities by derivatives

$$\nabla \mathbf{I} = \left[ \frac{\partial \mathbf{I}}{\partial x}, \frac{\partial \mathbf{I}}{\partial y} \right].$$

To only account for horizontal edges or vertical edges, we can represent changes in pixel intensities by  $[\frac{\partial \mathbf{I}}{\partial x}, 0]$  or  $[0, \frac{\partial \mathbf{I}}{\partial y}]$ , respectively. The gradient direction  $\theta$ , as in calculus, is

$$\theta = \tan^{-1} \left( \frac{\frac{\partial \mathbf{I}}{\partial y}}{\frac{\partial \mathbf{I}}{\partial x}} \right)$$

and the gradient strength is

$$\|\nabla \mathbf{I}\| = \sqrt{\left(\frac{\partial \mathbf{I}}{\partial x}\right)^2 + \left(\frac{\partial \mathbf{I}}{\partial y}\right)^2}.$$

In practice, however, images consist of discrete pixels, so we need a discrete approximation of image derivatives. Note that the definition of the derivative with respect to the  $x$ -coordinate is

$$\frac{\partial \mathbf{I}(x, y)}{\partial x} = \lim_{h \rightarrow 0} \frac{\mathbf{I}(x + h, y) - \mathbf{I}(x, y)}{h}$$

and its discrete counterpart is

$$\frac{\partial \mathbf{I}(x, y)}{\partial x} \approx \frac{\mathbf{I}(x + 1, y) - \mathbf{I}(x, y)}{1}.$$

To implement this derivative using a convolution, we can use a simple filter

$$F = \begin{bmatrix} -1 & 1 \end{bmatrix}.$$

However, a more commonly used 1D filter for edge detection is

$$F = \begin{bmatrix} -1 & 0 & 1 \end{bmatrix}$$

which corresponds to, in the continuous domain, a derivative definition that extends by a small amount in both directions:

$$\frac{\partial \mathbf{I}(x, y)}{\partial x} = \lim_{h \rightarrow 0} \frac{\mathbf{I}(x + h, y) - \mathbf{I}(x - h, y)}{2h}.$$

Note that the constant factor 2 is neglected in the filter since applying the unnormalized filter consistently will not affect relative intensity of the resulting image.

In the 2D case, three well known edge detection filters are the Prewitt, Sobel, and Roberts filters, which are displayed in the same order below.

$$\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$

Both the Prewitt and Sobel filters detect edges in the  $x$ -direction, and detection in the  $y$ -direction is similar but with the filter weights transposed. However, the Sobel filter is more commonly used between these two because it provides smoothing in the direction perpendicular to the direction of the edge detection. Details on smoothing will be provided in the next section. Images filtered by the Prewitt filter may also suffer from any noise in the original image. The Roberts filter detects edges in the diagonal direction instead. Fig. 3.7 shows an image filtered by each filter.



Figure 3.7: An image filtered by Prewitt, Sobel and Roberts filters, respectively.

### 3.3.3 Gaussian Filter

A Gaussian filter is used for smoothing and blurring. As indicated by its name, the filter resembles a Gaussian density function. Note that a multivariate Gaussian function centered at the origin can be written as

$$G(\vec{x} \mid \Sigma) = \frac{1}{\sqrt{(2\pi)^n |\Sigma|}} \exp \left\{ -\frac{1}{2} (\vec{x} - \vec{x}_0)^\top \Sigma^{-1} (\vec{x} - \vec{x}_0) \right\}$$

in which  $G$  denotes a Gaussian function,  $\vec{x}$  is an  $n$ -dimensional variable centered at  $\vec{x}_0$ , and  $\Sigma$  is the covariance matrix. However, for the purpose of image smoothing, a Gaussian filter is typically assumed to be independent among different coordinates. That is, in two dimensions, the filter can be written as

$$G(x, y \mid x_0, y_0, \sigma_x, \sigma_y) = \frac{1}{2\pi\sigma_x\sigma_y} e^{-((x-x_0)^2/2\sigma_x^2 + (y-y_0)^2/2\sigma_y^2)}$$

where by convention it is assumed that  $\sigma_x = \sigma_y = \sigma$ . Its discrete counterpart can be represented by filter

$$\frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}.$$

Varying  $\sigma$  allows one to vary the width of the Gaussian distribution, controlling the degree of smoothing. A larger  $\sigma$  corresponds to a larger filter size with each convolution accounting for more neighboring pixels. Note that Gaussian filtering is a form of a weighted sum with the highest weight at the center, effectively smoothing out noises in a close vicinity. A Gaussian-smoothed image is shown in Fig. 3.8.



Figure 3.8: An image smoothed by a Gaussian filter

### 3.3.4 Derivative of Gaussian & Laplacian of Gaussian Filter

Returning to edge detection, a major problem for Sobel and Prewitt filters is that they are very sensitive to noise due to a hard subtraction of neighboring values. It is therefore common to first conduct Gaussian smoothing before applying image gradients. Since convolution is associative, we can directly apply gradient operations to a Gaussian filter before applying them to the image itself.

This suggests that we need to calculate the first derivative of the Gaussian, and Fig. 3.9 shows the first derivatives, with respect to both the  $x$ - and  $y$ -coordinates, and their corresponding filters. Convolving a derivative of Gaussian filter with an image allows for both smoothing and edge detection, and it is less sensitive to noise than Prewitt or Sobel filters.

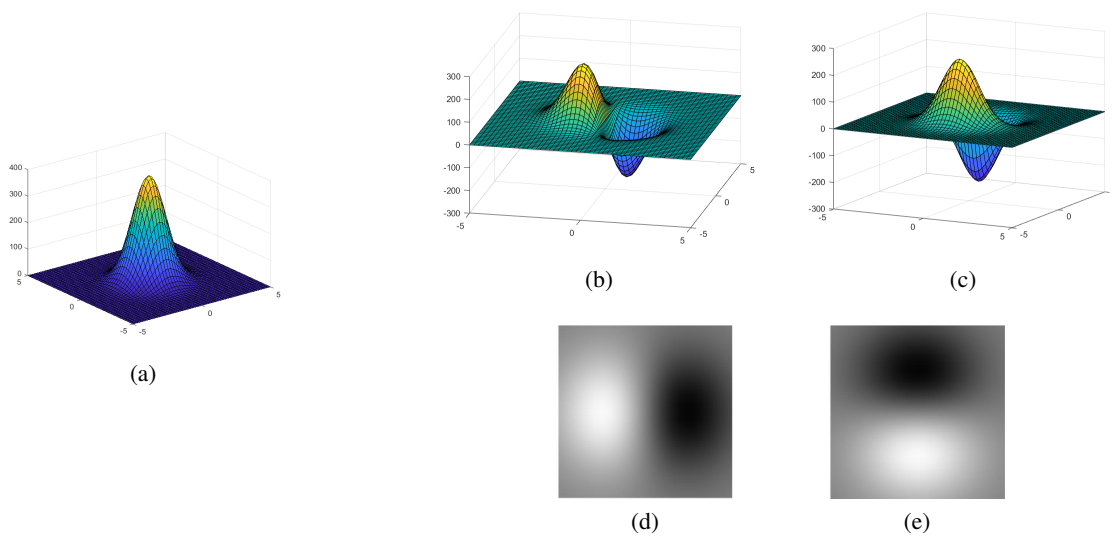


Figure 3.9: (a) Gaussian density (b) Density of  $x$ -derivative of Gaussian (c) Density of  $y$ -derivative of Gaussian (d) Filter visualization of  $x$ -derivative of Gaussian (e) Filter visualization of  $y$ -derivative of Gaussian

For edge detection purposes, first derivatives of the Gaussian are commonly used, but they are still somewhat undesirable due to the resulting thickness of detected edges, induced by  $\sigma$  in the Gaussian function. Second derivatives, on the other hand, represent edges with zero intensity, as the first derivative transforms edges to max/min intensity after filtering. This infinitely thin line of zero crossing is more desirable for edge detection. To calculate the second derivative, the Laplacian  $\Delta = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$  is most commonly used.

Similar to the derivation of image gradients, the second derivative in the  $x$  direction can be written as

$$\begin{aligned} \frac{\partial^2 \mathbf{I}(x, y)}{\partial x^2} &= \lim_{h \rightarrow 0} \frac{\frac{\partial \mathbf{I}(x+h, y)}{\partial x} - \frac{\partial \mathbf{I}(x-h, y)}{\partial x}}{2h} \\ &= \lim_{h \rightarrow 0} \frac{\lim_{t \rightarrow 0} \frac{\mathbf{I}(x+h, y) - \mathbf{I}(x+h-t, y)}{t} - \lim_{t \rightarrow 0} \frac{\mathbf{I}(x-h+t, y) - \mathbf{I}(x-h, y)}{t}}{2h} \\ &\approx \frac{\frac{\mathbf{I}(x+1, y) - \mathbf{I}(x, y)}{1} - \frac{\mathbf{I}(x, y) - \mathbf{I}(x-1, y)}{1}}{1} \\ &= \mathbf{I}(x+1, y) - 2\mathbf{I}(x, y) + \mathbf{I}(x-1, y). \end{aligned}$$

This corresponds to an (unnormalized) discrete filter

$$\begin{bmatrix} 1 & -2 & 1 \end{bmatrix}.$$

In 2D, it is not difficult to see that a simple Laplacian filter becomes

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}.$$

However, similar to image gradients, the Laplacian operator is extremely sensitive to noise in the input. Therefore, Gaussian smoothing is usually applied before a Laplacian operation. As the Laplacian can also be represented by a convolution as shown above, it can be applied to a Gaussian filter before being applied to the image. The resulting filter is named the Laplacian of Gaussian (LoG or LG). This gives a continuous formulation

$$\begin{aligned} LoG(x, y \mid x_0, y_0, \sigma_x, \sigma_y) &= \\ &= - \left[ \frac{1}{2\pi\sigma_x^3\sigma_y} \left[ 1 - \frac{(x-x_0)^2}{\sigma_x^2} \right] + \frac{1}{2\pi\sigma_x\sigma_y^3} \left[ 1 - \frac{(y-y_0)^2}{\sigma_y^2} \right] \right] \exp\left\{ -\left( \frac{(x-x_0)^2}{2\sigma_x^2} + \frac{(y-y_0)^2}{2\sigma_y^2} \right) \right\}. \end{aligned}$$

Fig. 3.10 shows both the probability density function of the Gaussian and the Laplacian of Gaussian. For the Laplacian of Gaussian, the uphill area surrounding a “valley” indicates that, by convolving it with an image, the low-intensity side of an edge in the image will rise to have high intensity before dropping to zero intensity and even lower, before then rising again to a high intensity on the other side of the edge. An example of filtering using the Laplacian of Gaussian is shown in Fig. 3.11.

### 3.3.5 Gabor Filter

Much effort has been spent on modeling radially-symmetric center-surround retinal ganglion cells. One such simple model for this purpose is the Laplacian of Gaussian introduced above with  $\sigma_x = \sigma_y = \sigma$ .

Yet, as retinal ganglion cells respond to specific frequency bands, Gabor filters also provide a good way to extract texture patterns from an image under specific frequencies [34]. In addition to being frequency-selective, Gabor filters are also directional-selective. Mathematically, a Gabor filter is a pair of cosine and

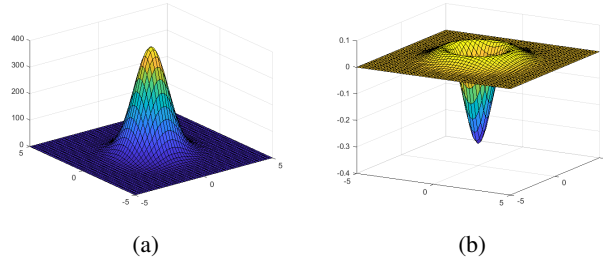


Figure 3.10: (a) Density plot of the Gaussian and (b) the second derivative of the Gaussian

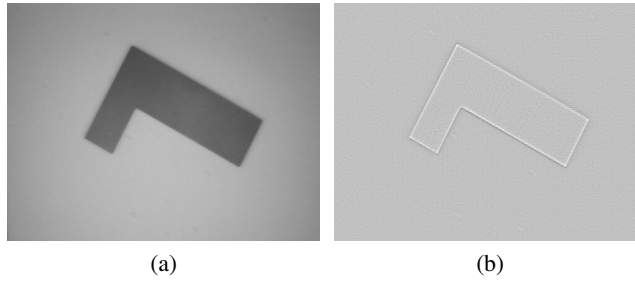


Figure 3.11: An example of filtering using the Laplacian of Gaussian

sine waves with angular frequency  $\omega$  and  $\eta$  respectively, and amplitude modulated by the Gaussian function. Its general form is

$$F_{\omega}(x, y) = G(x, y \mid x_0, y_0, \sigma_x, \sigma_y) \cdot S(x, y \mid \omega, \eta) \quad (3.12)$$

in which  $G(x, y \mid x_0, y_0, \sigma_x, \sigma_y)$  is a Gaussian function,  $S(x, y \mid \omega, \eta) = \exp\{-i(\omega x + \eta y)\}$  is a wave function, and  $\phi = \arctan(\eta/\omega)$  is the direction of the wave.

Here  $F_{\omega}$  defines a Gabor filter that matches image regions with frequency  $\omega$  in the  $x$ -direction and frequency  $\eta$  in the  $y$ -direction. High-frequency filters will match high-frequency patterns whereas low frequency filters will match low-frequency patterns.

Gabor filters are closely related to Gabor wavelets, which serve as bases for a Fourier Transform of an image. Some examples of Gabor wavelets are shown in Fig. 3.12.

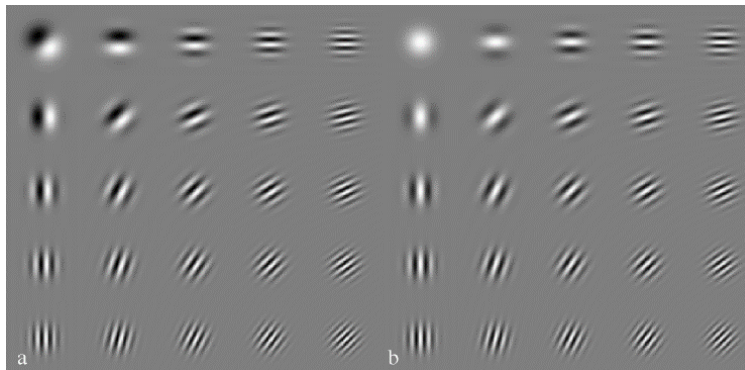


Figure 3.12: Gabor wavelets in different orientations and frequencies

The filters mentioned above are linear. Some functions are further applied to these linear filters to model the non-linear functions of the complex cell. One way to model the complex cell is to use the power of each pair of Gabor filters,  $|(F * \mathbf{I})(x, y)|^2$ . In fact,  $|(F * \mathbf{I})(x, y)|^2$  is the local spectrum of  $\mathbf{I}$  at  $(x, y)$  smoothed by a Gaussian function. Thus, it serves as a spectrum analyzer.

Although these filters are very efficient in capturing local spatial features, some problems are not well understood. First, given a bank of filters, how can we choose the best set of filters, especially when some are linear while others are non-linear, or the filters are highly correlated with each other? Second, after selecting the filters, how can we encapsulate the features they capture into a single texture model? These questions will be answered in the remainder of the chapter.

### 3.4 The FRAME Model

Markov Random Fields provide us physical inspiration for modeling texture patterns despite some notable limitations. Filters, on the other hand, afford a powerful and a biologically plausible way of extracting features. Is there a way to combine the two classes of texture models? In this coming section, we propose a modeling methodology which is built on and directly combines the above two important themes for texture modeling. It is called the FRAME (Filters, Random field, and Maximum Entropy) model [263]. Before continuing, however, you are encouraged to go over Chapter 9 and understand Maximum Entropy, Minimum Entropy, and Minimax Entropy Principles, as they will be employed in this chapter in derivations.

#### 3.4.1 Intuition and the Big Picture

Let image  $\mathbf{I}$  be defined on a domain  $\mathcal{D}$  where each  $\mathbf{I}_s \in \mathcal{L}$  is the intensity value of image  $\mathbf{I}$  at pixel  $s$ , and  $\mathcal{L}$  is an interval on the real line or a set of integers to represent pixel intensity. Without loss of generality, we denote a feature as  $\phi \in S = \{\phi^{(\alpha)}, \alpha = 1, 2, \dots, K\}$ , where  $K$  is the number of features in question. In the FRAME model, each feature is typically a vector of histograms of activations (normalized or unnormalized) that result from image  $\mathbf{I}$  being convolved with a selected filter. As a reminder, a filter can be seen as a low-dimensional space that pierces through the higher-dimensional image space. Accordingly, convolution of the image  $\mathbf{I}$  with the filter produces the projection of the image onto this low-dimensional space. The set of projected histograms are a set of marginal distributions for the image; our goal is to match “enough” marginal distributions that we can accurately model the observed image. An intuitive drawing is presented as Fig. 3.13.

Now given a set of observed images  $\{\mathbf{I}_i^{obs}, i = 1, 2, \dots, M\}$  from a distribution  $f(\mathbf{I})$ , we define feature statistics of the observation as

$$\mu_{obs}^{(\alpha)} = \frac{1}{M} \sum_{i=1}^M \phi^{(\alpha)}(\mathbf{I}_i^{obs}), \quad \text{for } \alpha = 1, \dots, K.$$

and the set of images that match the statistics of these features as

$$\Omega_{\mathbf{I}} = \{\mathbf{I} : \phi^{(\alpha)}(\mathbf{I}) = \mu_{obs}^{(\alpha)}, \alpha = 1, \dots, K\}.$$

Similarly, we can define the set of distributions that match the statistics as

$$\Omega_p = \{p(\mathbf{I}) : \mathbb{E}_p[\phi^{(\alpha)}(\mathbf{I})] = \mu_{obs}^{(\alpha)}, \alpha = 1, \dots, K\}.$$

We now have defined our feature statistics. The next task is to pursue a suitable distribution  $p \in \Omega_p$  that matches the observed distributions.

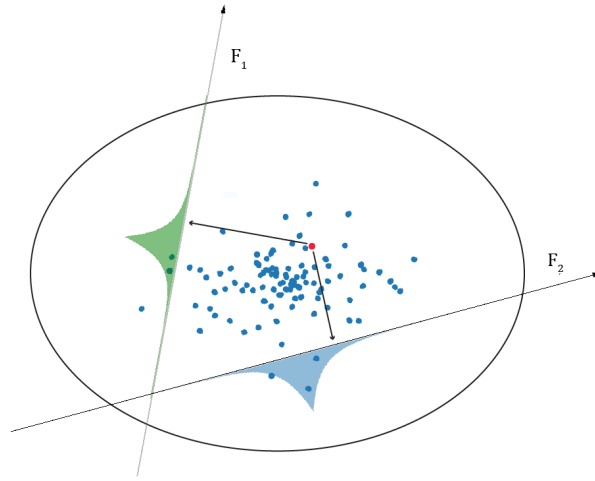


Figure 3.13: Suppose each point is an image and the red dot represents the selected image. Two filters are seen as axes piercing through the image space. The features described by these two filters are represented by the image's projection (i.e., convolution with a filter) against the two axes.

This pursuit process is iterative and two-fold. First, we select  $K$  suitable filters from a given set of filters to completely describe the most distinctive features of a texture. Second, we find the best parameterized distribution to explain the current statistics given a fixed set of filters. This parameterized distribution can be written as  $p(\mathbf{I}; \Lambda; S)$ , where  $\Lambda$  is the distribution parameters. Filter selection and distribution matching is an iterative process, reminiscent of coordinate descent, minimizing differences between the true distribution and the model distribution. Note that as we try to model pre-attentive processes (i.e., the subconscious accumulation and processing of environmental information) in human brains when detecting textures, we generally neglect training of filters and simply use pre-trained filters.

We make use of the Maximum Entropy Principle to pursue the best parameters  $\Lambda$  when the set of features (parameterized by a set of filters) to be matched is fixed. The parameterized distribution is constrained to match histograms of the filters' activation generated from texture images. However, we do not want to over-constrain our distribution since we should avoid adding extra information that may change the feature statistics. That is, we only match statistics given by the selected features and no more, meaning that we maximize entropy of our distribution while matching all the given features. More intuitively, we want to learn the distribution of the texture, but not to learn each training texture images precisely, so that we are able to generate new texture images that follow the same distribution as texture images in our training dataset but not reproduce the same images that are already in our training dataset. Fig. 3.14 illustrates this intuition.

To match the statistics provided by the observed data, an extreme solution would be a distribution that only concentrates on the data points themselves. However, this solution over-shrinks the constrained image set, adding too many unnecessary constraints, and it results in a highly overfitted model that can hardly generalize. Instead, we seek a distribution that exhibits maximum capacity under the current constraints. Remember in the earlier chapters, we discussed that the capacity of a space is measured by the entropy of the distribution defined for it. Thus, maximizing the entropy means finding the largest possible space.

As explained by the Maximum Entropy Principle, the solution to this constrained problem is in the form of a Gibbs distribution:

$$p(\mathbf{I}; \Lambda | S) = \frac{1}{Z(\Lambda)} \exp\left\{-\sum_{\alpha=1}^K \langle \lambda^{(\alpha)}, \phi^{(\alpha)}(\mathbf{I}) \rangle\right\}, \quad (3.13)$$

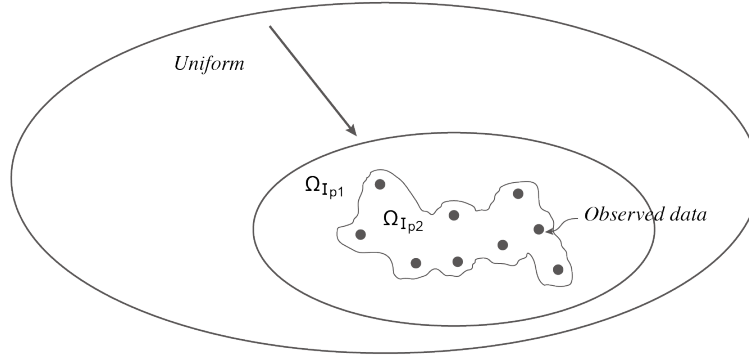


Figure 3.14: The Maximum Entropy Principle gives us the most general distribution that matches the feature statistics we choose. Here  $\Omega_{I_{p1}}$  and  $\Omega_{I_{p2}}$  are two possible distributions we learned to try to match the feature statistics of the observed data points.  $\Omega_{I_{p2}}$  concentrates too much to the observed data points which is representative of overfitting and poor generalization, while  $\Omega_{I_{p1}}$  is the more general one by applying the Maximum Entropy Principle.

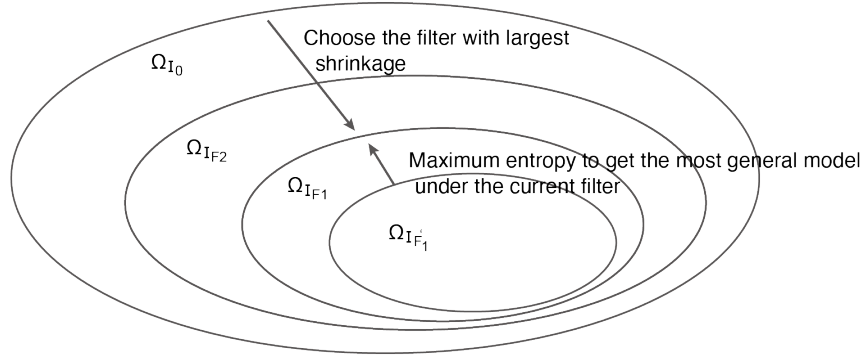


Figure 3.15: We can observe the min-max entropy process for training the FRAME model. Suppose we are at  $\Omega_{I_0}$  and we have two filters we could select:  $F_1$  and  $F_2$ . If either of them are selected, we update our model based on the Maximum Entropy principle, which lends the largest capacity based on the current constraint (e.g., preferring  $\Omega_{I_{F1}}$  to  $\Omega_{I_{F1}'}$ ). In choosing between  $F_1$  and  $F_2$ , we want the one that lends the largest reduction in the entropy of the parameterized distribution with respect to the feature space.

where  $\lambda^{(\alpha)}$  is the Lagrange multiplier associated with feature  $\phi^{(\alpha)}$  controlling the relative strength of activation,  $\langle \cdot, \cdot \rangle$  denotes inner product, and  $\Lambda = (\lambda^{(\alpha)}, \alpha = 1, \dots, K)$ .

The Lagrange multipliers are unique. After the current distribution is found, we proceed to find a new pre-trained filter to better model the texture pattern. Intuitively, we select the best filter that minimizes the Kullback-Leibler (KL) divergence from  $f(\mathbf{I})$  to  $p(\mathbf{I}; \Lambda, S)$ . With simplifications derived in the Minimum Entropy Principle section, minimizing this KL divergence is equivalent to minimizing entropy of the parameterized distribution with respect to the feature space.

This pursuit process can be visualized as below:

$$\text{Uniform} = p(\mathbf{I}; \Lambda_0, S_0) \rightarrow p(\mathbf{I}; \Lambda_1, S_1) \rightarrow \dots \rightarrow p(\mathbf{I}; \Lambda_K, S_K) \sim f(\mathbf{I})$$

The matched statistics give an equivalence class of images  $\Omega(\mathbf{h}) = \{\mathbf{I} : \mathbf{H}_i(\mathbf{I}) = \mathbf{h}_i, i = 1, \dots, K\}$  in which each  $\mathbf{I} \in \Omega(\mathbf{h})$  is similar in the sense that their projected histogram statistics are similar. Therefore,



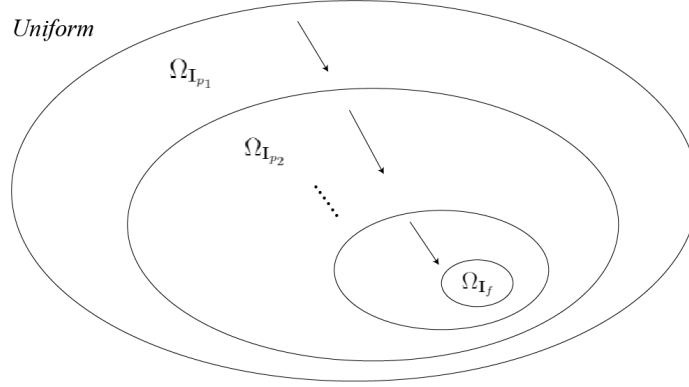


Figure 3.16: Reduction of constrained set of distributions from those sampled from uniform distribution, which corresponds to the entire pixel space, to an equivalence class of images that match projected statistics of the true samples. Here  $\Omega_{I_{p_1}}$  represents the set of images drawn from  $p_1$ , which follows a Gibbs distribution by the Maximum Entropy Principle. We perform this model pursuit process until images are close to images from  $f(\mathbf{I})$ , the true distribution.

the more filters are chosen, the more constrained the distributions are, and the fewer images there will be in the equivalence class of images. This model pursuit process is a process for gradually constraining the image space to obtain a set of images that match the true data distribution. The pursuit process is also shown in Fig. 3.16.

The figure can be interpreted as follows. We start from a random uniform distribution and start selecting a filter according to Minimum Entropy Principle. Then we use Maximum Entropy Principle to best match the selected feature statistics. This gives  $\Omega_{I_{p_1}}$  described by distribution  $p(\mathbf{I}; \Lambda_1, S_1)$ . The set of equivalence class of images also decreases, as indicated by the shrinking ellipse. This iterative process continues until our distribution is constrained enough so that it is indistinguishable from the observed distribution  $f(\mathbf{I})$ .

An interesting analogy to this pursuit process is shepherding. At first, we can imagine that sheep wander in an infinitely large area. At a certain time step, shepherds start guiding the sheep into a newly constructed fenced area, which defines a constrained space. Not wanting to be constrained further, the sheep push back against the fence, trying to escape. Shepherds then construct a new, smaller fenced area inside the last one and guide the sheep into this smaller area, with the sheep again pushing back. This process is iterated until the shepherds stop constructing fences.

Fig. 3.17 shows a toy example (images with 2 pixels) of this pursuit process. As shown,  $f(x, y)$  is the true data distribution we try to model, and our distribution  $p(x, y)$  starts from random uniform and a feature  $h_1$  is selected. Maximizing entropy of the constrained distribution gives us the second distribution. We further select the next best feature  $h_2$  and match our distribution against it. We can see that now the  $p(x, y)$  is already very close to  $f(x, y)$  and the process is stopped.

In the coming section, we derive the FRAME model and how it can be learned.

### 3.4.2 Deriving the FRAME model

With the bigger picture in mind, we now proceed to derive the FRAME model. We first fix a set of filters and try to learn  $\Lambda$ .

To re-iterate our definition, let image  $\mathbf{I}$  be defined on a discrete domain  $\mathcal{D}$ , in which  $\mathcal{D}$  can be a  $N \times N$  lattice. For each pixel  $s \in \mathcal{D}$ ,  $\mathbf{I}_s \in \mathcal{L}$ , and  $\mathcal{L}$  is an interval of  $\mathcal{R}$  or  $\mathcal{L} \subset \mathcal{Z}$ . For each texture, we assume that there exists a “true” joint probability density  $f(\mathbf{I})$  over the image space  $\mathcal{L}^{|\mathcal{D}|}$ , and  $f(\mathbf{I})$  should concentrate

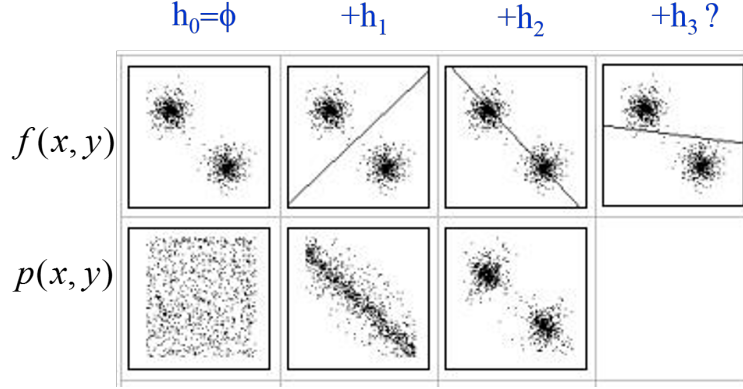


Figure 3.17: A toy example pursuit process as described by FRAME.

on a subspace of  $\mathcal{L}^{|\mathcal{D}|}$  (projection of  $\mathbf{I}$  onto the subspace) which corresponds to texture images that have perceptually similar texture appearances.

### Learning Potential Functions

Given an image  $\mathbf{I}$  and a filter  $F^{(\alpha)}$  with  $\alpha = 1, 2, \dots, K$  for the indices of the filter, we let  $\mathbf{I}_s^{(\alpha)} = F^{(\alpha)} * \mathbf{I}_s$  be the filter response at location  $s$  and  $\mathbf{I}^{(\alpha)}$  the filtered image. The marginal empirical distribution (histogram) of  $\mathbf{I}^{(\alpha)}$  is

$$H^{(\alpha)}(z) = \frac{1}{|\mathcal{D}|} \sum_{s \in \mathcal{D}} \delta(z - \mathbf{I}_s^{(\alpha)}),$$

where  $z$  is a specific filter response and  $\delta(\cdot)$  is the Dirac delta function. The marginal distribution of  $f(\mathbf{I})$  with respect to  $F^{(\alpha)}$  at location  $s$  is denoted by

$$f_s^{(\alpha)}(z) = \int_{\mathbf{I}_s^{(\alpha)}=z} f(\mathbf{I}) d\mathbf{I} = E_f[\delta(z - \mathbf{I}_s^{(\alpha)})].$$

At first thought, it seems an intractable problem to estimate  $f(\mathbf{I})$  due to the overwhelming dimensionality of image  $\mathbf{I}$ . To reduce dimensions, we first introduce the following theorem.

**Theorem 1.** *Let  $f(\mathbf{I})$  be the  $|\mathcal{D}|$ -dimensional continuous probability distribution of a texture, then  $f(\mathbf{I})$  is a linear combination of  $f^{(\xi)}$ , the latter are the marginal distributions on the linear filter response  $F^{(\xi)} * \mathbf{I}$ .*

*Proof.* By inverse Fourier transform, we have

$$f(\mathbf{I}) = \frac{1}{(2\pi)^{|\mathcal{D}|}} \int e^{2\pi i \langle \mathbf{I}, \xi \rangle} \hat{f}(\xi) d\xi,$$

where  $\hat{f}(\xi)$  is the characteristic function of  $f(\mathbf{I})$  and

$$\begin{aligned} \hat{f}(\xi) &= \int e^{-2\pi i \langle \xi, \mathbf{I} \rangle} f(\mathbf{I}) d\mathbf{I} \\ &= \int e^{-2\pi i z} \int_{\langle \xi, \mathbf{I} \rangle = z} f(\mathbf{I}) dz d\mathbf{I} \end{aligned}$$

$$\begin{aligned}
&= \int e^{-2\pi iz} \int \delta(z - \langle \xi, \mathbf{I} \rangle) f(\mathbf{I}) d\mathbf{I} dz \\
&= \int e^{-2\pi iz} f^{(\xi)}(z) dz,
\end{aligned}$$

where  $\langle \cdot, \cdot \rangle$  is the inner product, by definition  $f^{(\xi)}(z) = \int \delta(z - \langle \xi, \mathbf{I} \rangle) f(\mathbf{I}) d\mathbf{I}$  is the marginal distribution of  $F^{(\xi)} * \mathbf{I}$ , and we define  $F^{(\xi)}(s) = \xi(s)$  as a linear filter.  $\square$

Theorem 1 transforms  $f(\mathbf{I})$  into a linear combination of its one-dimensional marginal distributions. Thus, it motivates a new method for inferring  $f(\mathbf{I})$ ; namely, construct a distribution  $p(\mathbf{I})$  so that  $p(\mathbf{I})$  has the same marginal distributions  $f^{(\xi)}$ . If  $p(\mathbf{I})$  matches all marginal distributions of  $f(\mathbf{I})$ , then  $p(\mathbf{I}) = f(\mathbf{I})$ . But this method will involve an uncountable number of filters, and each filter  $F^{(\xi)}$  is as large as image  $\mathbf{I}$ .

Our second motivation comes from psychophysical research on human texture perception, which suggests that two homogeneous textures are often difficult to discriminate when they produce similar *marginal distributions* for responses from a *bank of filters* [16]. This means that it is plausible to ignore some statistical properties of  $f(\mathbf{I})$  which are not important for human texture discrimination.

To make texture modeling a tractable problem, we make the following assumptions to limit the number of filters and the window size of each filter for computational reasons, though these assumptions are not necessary conditions for our theory to hold true. First, we limit our model to homogeneous textures; thus,  $f(\mathbf{I})$  is stationary with respect to location  $s$ . Second, for a given texture, all features which concern texture perception can be captured by “locally” supported filters. In other words, the sizes of filters should be smaller than the size of the image. For example, the size of the image is  $256 \times 256$  pixels, and the sizes of filters we used are limited to less than  $33 \times 33$  pixels. These filters can be linear or non-linear as we discussed in Section 3.3. Third, only a finite set of filters are used to estimate  $f(\mathbf{I})$ .

The first and second assumptions are made because we often have access to only one observed (training) texture image. For a given observed image  $\mathbf{I}^{obs}$  and a filter  $F^{(\alpha)}$ , we let  $\mathbf{I}^{obs(\alpha)}$  denote the filtered image, and  $H^{obs(\alpha)}(z)$  the histogram of  $\mathbf{I}^{obs(\alpha)}$ . According to the first assumption,  $f_s^{(\alpha)}(z) = f^{(\alpha)}(z)$  is independent of  $s$ . By ergodicity,  $H^{obs(\alpha)}(z)$  makes a consistent estimator of  $f^{(\alpha)}(z)$ . The second assumption ensures that the image size is larger relative to the support of filters, so that ergodicity takes effect for  $H^{obs(\alpha)}(z)$  to be an accurate estimate of  $f^{(\alpha)}(z)$ .

Now for a specific texture, let  $S_K = \{F^{(\alpha)}, \alpha = 1, \dots, K\}$  be a finite set of well selected filters and  $f^{(\alpha)}(z), \alpha = 1, \dots, K$  be the corresponding marginal distributions of  $f(\mathbf{I})$ . We denote the probability distribution  $p(\mathbf{I})$  that matches these marginal distributions as a set

$$\Omega_{p_K} = \{p(\mathbf{I}) \mid \mathbb{E}_p[\delta(z - \mathbf{I}_s^{(\alpha)})] = f^{(\alpha)}(z) \quad \forall z \in R, \quad \forall \alpha = 1, \dots, K, \quad \forall s \in \mathcal{D}\}, \quad (3.14)$$

where  $\mathbb{E}_p[\delta(z - \mathbf{I}_s^{(\alpha)})]$  is the marginal distribution of  $p(\mathbf{I})$  with respect to filter  $F^{(\alpha)}$  at location  $s$ . Thus according to the third assumption, any  $p(\mathbf{I}) \in \Omega_{p_K}$  is perceptually an adequate model for the texture, provided that we have enough well-selected filters. Then we choose from  $\Omega_{p_K}$  a distribution  $p(\mathbf{I})$  which has the maximum entropy:

$$\begin{aligned}
p(\mathbf{I}) &= \arg \max \left\{ - \int p(\mathbf{I}) \log p(\mathbf{I}) d\mathbf{I} \right\}, \\
\text{subject to} \quad & \mathbb{E}_p[\delta(z - \mathbf{I}_s^{(\alpha)})] = f^{(\alpha)}(z), \quad \forall z \in R, \quad \forall \alpha = 1, \dots, K, \quad \forall s \in \mathcal{D}, \\
\text{and} \quad & \int p(\mathbf{I}) d\mathbf{I} = 1.
\end{aligned} \quad (3.15)$$

We can achieve the maximum entropy distribution by making  $p(\mathbf{I})$ , while it satisfies constraints in some dimensions, as random as possible in the other unconstrained dimensions. This makes it such that  $p(\mathbf{I})$

is the most universal set it could be given its constraints; it does not capture, or represent, any additional constraints that are not explicitly defined by  $p(\mathbf{I})$ . In this case, by not representing more information than we have available, we say the ME distribution exhibits the purest fusion of the extracted features.

Using Lagrange multipliers, solving the constrained optimization problem in Equation 3.15 is equal to solving the underlying unconstrained problem:

$$\begin{aligned}
p(\mathbf{I}) &= \arg \max \left\{ - \int p(\mathbf{I}) \log p(\mathbf{I}) d\mathbf{I} \right. \\
&\quad \left. - \sum_s \sum_{\alpha=1}^K \int \lambda^{(\alpha)}(z) (E_p[\delta(z - \mathbf{I}_s^{(\alpha)})] - f^{(\alpha)}(z)) dz - \lambda^{(\beta)} \left( \int p(\mathbf{I}) d\mathbf{I} - 1 \right) \right\} \\
&= \arg \max \left\{ - \int p(\mathbf{I}) \log p(\mathbf{I}) d\mathbf{I} \right. \\
&\quad \left. - \sum_s \sum_{\alpha=1}^K \int \lambda^{(\alpha)}(z) \left( \int \delta(z - \mathbf{I}_s^{(\alpha)}) p(\mathbf{I}) d\mathbf{I} - f^{(\alpha)}(z) \right) dz - \lambda^{(\beta)} \left( \int p(\mathbf{I}) d\mathbf{I} - 1 \right) \right\} \quad (3.16)
\end{aligned}$$

where  $\lambda^{(\alpha)}(z)$   $\alpha = 1, 2, \dots, K$  and  $\lambda^{(\beta)}$  are Lagrange multipliers we introduce. Note that because the constraints on Equation 3.15 differ from the ones given in Section 3.4.1 in that  $z$  takes continuous real values and there is an uncountable number of constraints. Therefore, the Lagrange parameter  $\lambda$  takes the form as a function of  $z$ . Also, since the constraints are the same for all locations  $s \in \mathcal{D}$ ,  $\lambda$  should be independent of  $s$ . According the Euler-Lagrange equation, if we want to find the stationary point (maximum or minimum) for the functional

$$S(f(x)) = \int_a^b L(x, f(x), \dot{f}(x)) dx,$$

then the function  $f(x)$  should satisfy

$$\frac{\partial L}{\partial f} - \frac{d}{dx} \frac{\partial L}{\partial \dot{f}} = 0.$$

Applying this to Equation 3.16, we can derive that the optimum  $P(I)$  should satisfy

$$\begin{aligned}
& -\log p(\mathbf{I}) - 1 - \sum_s \sum_{\alpha=1}^K \int \lambda^{(\alpha)}(z) \delta(z - \mathbf{I}_s^{(\alpha)}) dz - \lambda^{(\beta)} = 0 \\
& p(\mathbf{I}) = \exp \left\{ -1 - \lambda^{(\beta)} - \sum_s \sum_{\alpha=1}^K \int \lambda^{(\alpha)}(z) \delta(z - \mathbf{I}_s^{(\alpha)}) dz \right\}. \quad (3.17)
\end{aligned}$$

By reorganizing Equation 3.17, we can get the following ME distribution:

$$p(\mathbf{I}; \Lambda_K, S_K) = \frac{1}{Z(\Lambda_K)} \exp \left\{ - \sum_s \sum_{\alpha=1}^K \int \lambda^{(\alpha)}(z) \delta(z - \mathbf{I}_s^{(\alpha)}) dz \right\} \quad (3.18)$$

$$= \frac{1}{Z(\Lambda_K)} \exp \left\{ - \sum_s \sum_{\alpha=1}^K \lambda^{(\alpha)}(\mathbf{I}_s^{(\alpha)}) \right\} \quad (3.19)$$

where  $S_K = \{F^{(1)}, F^{(2)}, \dots, F^{(K)}\}$  is a set of selected filters and  $\Lambda_K = (\lambda^{(1)}(\cdot), \lambda^{(2)}(\cdot), \dots, \lambda^{(K)}(\cdot))$  is the Lagrange parameter.  $Z(\Lambda_K)$  is the normalizing constant that contains the term  $\exp\{-1 - \lambda^{(\beta)}\}$ . Note that

in Equation 3.19, for each filter  $F^{(\alpha)}$ ,  $\lambda^{(\alpha)}(\cdot)$  takes the form as a continuous function of the filter response  $\mathbf{I}_s^{(\alpha)}$ .

To proceed further, let us derive a discrete form of Equation 3.19. Assume that the filter response  $\mathbf{I}_s^{(\alpha)}$  is discretized into  $L$  discrete gray levels, and as such  $z$  takes values from  $\{z_1^{(\alpha)}, z_2^{(\alpha)}, \dots, z_L^{(\alpha)}\}$ . In general, the width of these bins do not have to be equal, and the number of gray levels  $L$  for each filter response may vary. As a result, the marginal distributions and histograms are approximated by piecewise constant functions of  $L$  bins, and we denote these piecewise functions as vectors.  $H^{(\alpha)} = (H_1^{(\alpha)}, H_2^{(\alpha)}, \dots, H_L^{(\alpha)})$  is the histogram of  $\mathbf{I}^{(\alpha)}$ ,  $H^{obs(\alpha)}$  denotes the histogram of  $\mathbf{I}^{obs(\alpha)}$ , and the potential function  $\lambda^{(\alpha)}(\cdot)$  is approximated by vector  $\lambda^{(\alpha)} = (\lambda_1^{(\alpha)}, \lambda_2^{(\alpha)}, \dots, \lambda_L^{(\alpha)})$ .

So Equation 3.18 is rewritten as

$$p(\mathbf{I}; \Lambda_K, S_K) = \frac{1}{Z(\Lambda_K)} \exp\left\{-\sum_s \sum_{\alpha=1}^K \sum_{i=1}^L \lambda_i^{(\alpha)} \delta(z_i^{(\alpha)} - \mathbf{I}_s^{(\alpha)})\right\},$$

and by changing the order of summations:

$$\begin{aligned} p(\mathbf{I}; \Lambda_K, S_K) &= \frac{1}{Z(\Lambda_K)} \exp\left\{-\sum_{\alpha=1}^K \sum_{i=1}^L \lambda_i^{(\alpha)} H_i^{(\alpha)}\right\} \\ &= \frac{1}{Z(\Lambda_K)} \exp\left\{-\sum_{\alpha=1}^K \langle \lambda^{(\alpha)}, H^{(\alpha)} \rangle\right\}. \end{aligned} \quad (3.20)$$

The virtue of Equation 3.20 is that it provides us with a simple parametric model for the probability distribution on  $\mathbf{I}$ , and this model has the following properties:

- $p(\mathbf{I}; \Lambda_K, S_K)$  is specified by  $\Lambda_K = (\lambda^{(1)}, \lambda^{(2)}, \dots, \lambda^{(K)})$  and  $S_K$ .
- Given an image  $\mathbf{I}$ , its histograms  $H^{(1)}, H^{(2)}, \dots, H^{(K)}$  are sufficient statistics, i.e.,  $p(\mathbf{I}; \Lambda_K, S_K)$  is a function of  $(H^{(1)}, H^{(2)}, \dots, H^{(K)})$ .

We plug Equation 3.20 into the constraints of the ME distribution and solve for  $\lambda^{(\alpha)}, \alpha = 1, 2, \dots, K$  iteratively by the following equations:

$$\frac{d\lambda^{(\alpha)}}{dt} = E_{p(\mathbf{I}; \Lambda_K, S_K)}[H^{(\alpha)}] - H^{obs(\alpha)}. \quad (3.21)$$

In Equation 3.21, we have substituted  $H^{obs(\alpha)}$  for  $f^{(\alpha)}$ , and  $E_{p(\mathbf{I}; \Lambda_K, S_K)}(H^{(\alpha)})$  is the expected histogram of the filtered image  $\mathbf{I}^{(\alpha)}$  where  $\mathbf{I}$  follows  $p(\mathbf{I}; \Lambda_K, S_K)$  with the current  $\Lambda_K$ . We converge to the unique solution at  $\Lambda_K = \hat{\Lambda}_K$ , and  $\hat{\Lambda}_K$  is called the ME-estimator.

It is worth mentioning that this ME-estimator is equivalent to the maximum likelihood estimator (MLE):

$$\begin{aligned} \hat{\Lambda}_K &= \arg \max_{\Lambda_K} \log p(\mathbf{I}^{obs}; \Lambda_K, S_K) \\ &= \arg \max_{\Lambda_K} -\log Z(\Lambda_K) - \sum_{\alpha=1}^K \langle \lambda^{(\alpha)}, H^{obs(\alpha)} \rangle. \end{aligned} \quad (3.22)$$

By gradient ascent, maximizing the log-likelihood lends Equation 3.21, following property i) of the partition function  $Z(\Lambda_K)$ . In Equation 3.21, at each step, given  $\Lambda_K$  and hence  $p(\mathbf{I}; \Lambda_K, S_K)$ , the analytic

form of  $E_{p(\mathbf{I}; \Lambda_K, S_K)}(H^{(\alpha)})$  is not available; instead, we propose the following method to estimate it, as we did for  $f^{(\alpha)}$  before. We draw a typical sample from  $p(\mathbf{I}; \Lambda_K, S_K)$  and thus synthesize a texture image  $\mathbf{I}^{syn}$ . Then we use the histogram  $H^{syn(\alpha)}$  of  $\mathbf{I}^{syn(\alpha)}$  to approximate  $E_{p(\mathbf{I}; \Lambda_K, S_K)}(H^{(\alpha)})$ . This requires that the size of  $\mathbf{I}^{syn}$  that we are synthesizing should be large enough.

To draw a typical sample image from  $p(\mathbf{I}; \Lambda_K, S_K)$ , we use the Gibbs sampler which simulates a Markov chain in the image space  $\mathcal{L}^{|\mathcal{D}|}$ . The Markov chain starts from any random image, e.g., a white noise image, and it converges to a stationary process with distribution  $p(\mathbf{I}; \Lambda_K, S_K)$ . Thus, when the Gibbs sampler converges, the images synthesized follow distribution  $p(\mathbf{I}; \Lambda_K, S_K)$ .

In summary, we propose Algorithm 1 for inferring the underlying probability model  $p(\mathbf{I}; \Lambda_K, S_K)$  and for synthesizing the texture according to  $p(\mathbf{I}; \Lambda_K, S_K)$ . The algorithm stops when the sub-band histograms of the synthesized texture closely match the corresponding histograms of the observed images.\*.

Input a texture image  $\mathbf{I}^{obs}$ .  
 Select a group of  $K$  filters  $S_K = \{F^{(1)}, F^{(2)}, \dots, F^{(K)}\}$ .  
 Compute  $\{H^{obs(\alpha)}, \alpha = 1, \dots, K\}$ .  
 Initialize  $\lambda_i^{(\alpha)} \leftarrow 0, i = 1, 2, \dots, L, \alpha = 1, 2, \dots, K$ .  
 Initialize  $\mathbf{I}^{syn}$  as a uniform white noise texture.<sup>†</sup>  
 Repeat  
   Calculate  $H^{syn(\alpha)} \alpha = 1, 2, \dots, K$  from  $\mathbf{I}^{syn}$ , use it for  $E_{p(\mathbf{I}; \Lambda_K, S_K)}(H^{(\alpha)})$ .  
   Update  $\lambda^{(\alpha)} \alpha = 1, 2, \dots, K$  by Equation 3.21), so  $p(\mathbf{I}; \Lambda_K, S_K)$  is updated.  
   Apply Gibbs sampler to flip  $\mathbf{I}^{syn}$  for  $w$  sweeps under  $p(\mathbf{I}; \Lambda_K, S_K)$   
 Until  $\frac{1}{2} \sum_i^L |H_i^{obs(\alpha)} - H_i^{syn(\alpha)}| \leq \epsilon$  for  $\alpha = 1, 2, \dots, K$ .

### Algorithm 1: The learning algorithm

Given image  $\mathbf{I}_s$ , flip\_counter  $\leftarrow 0$   
 Repeat  
   Randomly pick a location  $s$  under the uniform distribution.  
   For  $val = 0, \dots, G - 1$  with  $G$  being the number of grey levels of  $\mathbf{I}$   
     Calculate  $p(\mathbf{I}_s = val | \mathbf{I}_{-s})$  by  $p(\mathbf{I}; \Lambda_K, S_K)$ .  
     Randomly flip  $\mathbf{I}_s \leftarrow val$  under  $p(val | \mathbf{I}_{-s})$ .  
     flip\_counter  $\leftarrow$  flip\_counter + 1  
 Until flip\_counter =  $w \times M \times N$ .

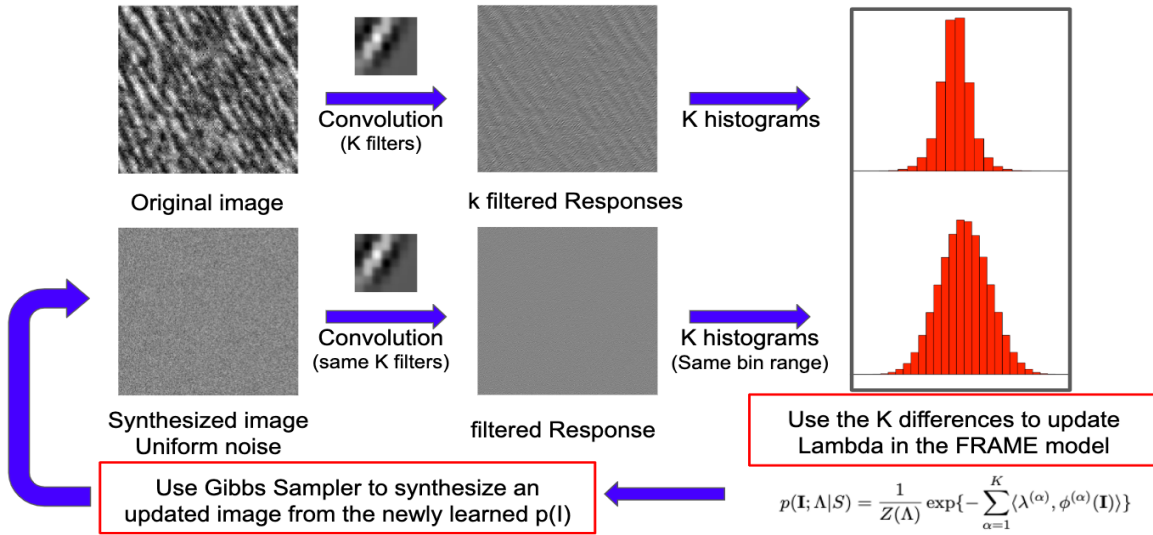
### Algorithm 2: The Gibbs Sampler for $w$ sweeps

In Algorithm 2, to compute  $p(\mathbf{I}_s = val | \mathbf{I}_{-s})$ , we set  $\mathbf{I}_s$  to  $val$ , and due to the Markov property we only need to compute the changes of  $\mathbf{I}^{(\alpha)}$  in the neighborhood of  $s$ . The size of the neighborhood is determined

---

\*We assume the histogram of each sub-band  $\mathbf{I}^{(\alpha)}$  is normalized such that  $\sum_i H_i^{(\alpha)} = 1$ , therefore all the  $\{\lambda_i^{(\alpha)}, i = 1, \dots, L\}$  computed in this algorithm have one extra degree of freedom for each  $\alpha$ , i.e., we can increase  $\{\lambda_i^{(\alpha)}, i = 1, \dots, L\}$  by a constant without changing  $p(\mathbf{I}; \Lambda_K, S_K)$ . This constant will be absorbed by the partition function  $Z(\Lambda_K)$ .

<sup>†</sup>The white noise image with uniform distribution are the samples from  $p(\mathbf{I}; \Lambda_K, S_K)$  with  $\lambda_i^{(\alpha)} = 0$



Stop until the 2 histograms look the same for each K filters. If a model can synthesize an image that has K similar filtered responses as those of original image, then the synthesized image is similar to the original image.

Figure 3.18: **Algorithm 1:** Given a texture image and K filters (after filter selection), perform convolution to extract K histograms from the filtered response, denoting each as  $h^{obs(\alpha)}$ . Synthesize an image (initialized as white noise) that is same size of the input image, and perform convolution to extract another K histograms from the filtered response, denoting each as  $h^{syn(\alpha)}$ . Update the coefficient of the FRAME model  $\lambda$  by  $\lambda_{t+1}^\alpha = \lambda_t^\alpha + \eta(h^{syn(\alpha)} - h^{obs(\alpha)})$ . With this new model, we can again use the Gibbs sampler for W sweeps to synthesize another image for the next iteration.

by the size of filter  $F^{(\alpha)}$ . With the updated  $\mathbf{I}^{(\alpha)}$ , we calculate  $H^{(\alpha)}$ , and the probability is normalized such that  $\sum_{val=0}^{G-1} p(\mathbf{I}_s = val \mid \mathbf{I}_{-s}) = 1$ .

In the Gibbs sampler, flipping a pixel is a step of the Markov chain, and we define flipping  $|\mathcal{D}|$  pixels as a sweep, where  $|\mathcal{D}|$  is the size of the synthesized image. Then the overall iterative process becomes an inhomogeneous Markov chain. At the beginning of the process,  $p(\mathbf{I}; \Lambda_K, S_K)$  is a ‘‘hot’’ uniform distribution. By updating the parameters, the process gets closer and closer to the target distribution, which is much colder. So the algorithm is very much like a simulated annealing algorithm, which is helpful for getting around local modes of the target distribution.

In summary, the FRAME model incorporates and generalizes the attractive properties of filtering theories and the random field models. Moreover, it interprets many previous methods for texture modeling with a unifying perspective.

### Filter Selection

After  $\Lambda$  is learned with the above algorithm, we can now proceed to select the next filter to be added to the set. One way to choose  $S_K$  from a filter bank  $\mathcal{B}$  is to search for all possible combinations of  $K$  filters in  $\mathcal{B}$  and compute the corresponding  $p(\mathbf{I}; \Lambda_K, S_K)$ . Then by comparing the synthesized texture images following each  $p(\mathbf{I}; \Lambda_K, S_K)$ , we can see which set of filters is the best. Such a brute force search is computationally infeasible, and for a specific texture we often do not know what  $K$  is. Instead, we propose a stepwise greedy strategy. We start from  $S_0 = \emptyset$  (hence  $p(\mathbf{I}; \Lambda_0, S_0)$  is a uniform distribution), and then sequentially introduce one filter at a time. Namely, we want to find a filter that reduces  $\text{KL}(f \parallel p(\mathbf{I}; \Lambda))$  the most (by the minimum entropy principle).

Suppose that at the  $k$ -th step we have chosen  $S_k = \{F^{(1)}, F^{(2)}, \dots, F^{(k)}\}$  and obtained a maximum entropy distribution

$$p(\mathbf{I}; \Lambda_k, S_k) = \frac{1}{Z(\Lambda_k)} \exp\left\{-\sum_{\alpha=1}^k \langle \lambda^{(\alpha)}, H^{(\alpha)} \rangle\right\}, \quad (3.23)$$

so that  $E_{p(\mathbf{I}; \Lambda_k, S_k)}[H^{(\alpha)}] = f^{(\alpha)}$  for  $\alpha = 1, 2, \dots, k$ . Then at the  $(k+1)$ -th step, for each filter  $F^{(\beta)} \in \mathcal{B}/S_k$ , we denote by  $d(\beta) = D(E_{p(\mathbf{I}; \Lambda_k, S_k)}[H^{(\beta)}], f^{(\beta)})$  the distance between  $E_{p(\mathbf{I}; \Lambda_k, S_k)}[H^{(\beta)}]$  and  $f^{(\beta)}$ , which are respectively the marginal distributions of  $p(\mathbf{I}; \Lambda_k, S_k)$  and  $f(\mathbf{I})$  with respect to filter  $F^{(\beta)}$ . Intuitively, the bigger  $d(\beta)$  is, the more information  $F^{(\beta)}$  carries, since it reports a big difference between  $p(\mathbf{I}; \Lambda_k, S_k)$  and  $f(\mathbf{I})$ . Therefore we should choose the filter which has the maximal distance, i.e.,

$$F^{k+1} = \arg \max_{F^{(\beta)} \in \mathcal{B}/S_k} D(E_{p(\mathbf{I}; \Lambda_k, S_k)}[H^{(\beta)}], f^{(\beta)}). \quad (3.24)$$

To minimize entropy, we naturally choose the next filter to be the one that maximizes reduction in  $\text{KL}(f \parallel p(\mathbf{I}; \Lambda))$ . This indicates that we want to minimize

$$D(E_{p(\mathbf{I}; \Lambda_{k+1}, S_{k+1})}[H^{(\beta)}], f^{(\beta)}) = \text{KL}(f \parallel p(\mathbf{I}; \Lambda_k, S_k)) - \text{KL}(f \parallel p(\mathbf{I}; \Lambda_{k+1}, S_{k+1})). \quad (3.25)$$

Note that the above difference is a distance measure since

$$\text{KL}(f \parallel p(\mathbf{I}; \Lambda_k, S_k)) - \text{KL}(f \parallel p(\mathbf{I}; \Lambda_{k+1}, S_{k+1})) = \text{KL}(p(\mathbf{I}; \Lambda_{k+1}, S_{k+1}) \parallel p(\mathbf{I}; \Lambda_k, S_k)). \quad (3.26)$$

**Proposition 1.** *Using the distance metric above,  $\text{KL}(f \parallel p(\mathbf{I}; \Lambda_k, S_k)) - \text{KL}(f \parallel p(\mathbf{I}; \Lambda_{k+1}, S_{k+1}))$  is equivalent to information gain:  $\text{entropy}(p(\mathbf{I}; \Lambda_k, S_k)) - \text{entropy}(p(\mathbf{I}; \Lambda_{k+1}, S_{k+1}))$ .*



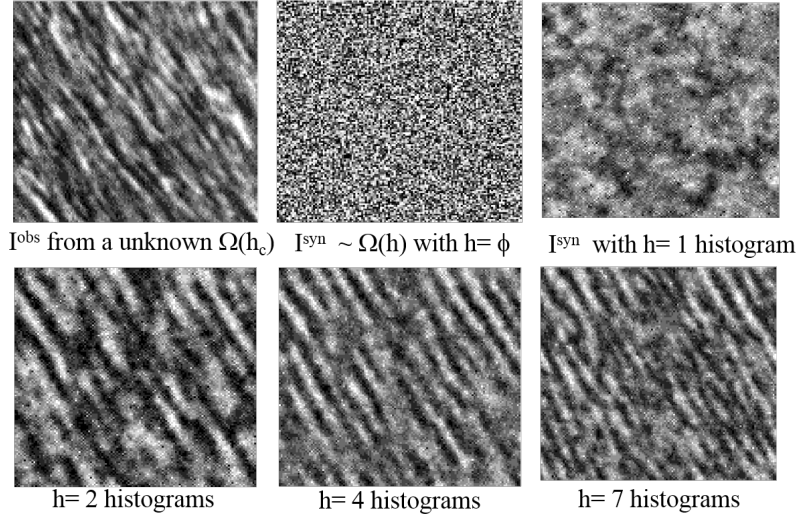


Figure 3.19: Synthesis of a fur texture: a is the observed texture; b,c,d,e,f are the synthesized textures using  $K = 0, 1, 2, 4, 7$  filters, respectively.

*Proof.* To simplify notation, we denote  $p(\mathbf{I}; \Lambda_k, S_k)$  as  $p$  and  $p(\mathbf{I}; \Lambda_{k+1}, S_{k+1})$  as  $p_+$ :

$$\begin{aligned} \text{KL}(f||p) - \text{KL}(f||p_+) &= \text{entropy}(f) - \mathbb{E}_f[\log p] - \text{entropy}(f) + \mathbb{E}_f[\log p_+] \\ &= \mathbb{E}_f[\log p_+] - \mathbb{E}_f[\log p] \end{aligned}$$

We refer to Theorem 1 of Section 9, which shows that  $\mathbb{E}_f[\log p] = -\text{entropy}(p)$  and  $\mathbb{E}_f[\log p_+] = -\text{entropy}(p_+)$ . Then  $\mathbb{E}_f[\log p_+] - \mathbb{E}_f[\log p] = \text{entropy}(p) - \text{entropy}(p_+)$ , which is the definition of information gain. Although we get the expression of information gain, it is not directly computable. We need to approximate it with the help of Taylor expansion.  $\square$

**Proposition 2.** Simplifying the distance measure as  $D(h)$ , where  $h$  is the evaluated histogram feature using the newly added filter,  $L_p$ -norm is a reasonable approximation to  $D(h)$ .

*Proof.* Denote  $p$  as the distribution modeled prior to adding the new filter, then let  $h_0$  be the feature evaluated using the new filter on samples from  $p$  and  $h^{obs}$  be the feature evaluated using the new filter on samples from true distribution  $f$ . Using Taylor expansion,

$$D(h^{obs}) \approx D(h) \Big|_{h=h_0} + \frac{\partial D}{\partial h} \Big|_{h=h_0} (h^{obs} - h_0) + \frac{1}{2} \frac{\partial^2 D}{\partial h^2} \Big|_{h=h_0} (h^{obs} - h_0)^2$$

where  $\frac{\partial D}{\partial h} = -\lambda^{k+1}$  and  $\frac{\partial^2 D}{\partial h^2}$  is the covariance matrix  $\Sigma_{h^{obs}}$  of each component of  $h^{obs}$  evaluated using the new filter (for a detailed explanation, please refer to Chapter 9). As  $\lambda^{k+1}$  is typically initialized to 0, we also have  $D(h) \Big|_{h=h_0} = 0$  and  $\frac{\partial D}{\partial h} \Big|_{h=h_0} (h^{obs} - h_0) = 0$ .

Furthermore, since we assume each component is independent from each other, we can approximate the second order term by  $L_2$  distance. In general,  $L_p$  distance metric is a good enough approximation to  $D(h)$ .  $\square$

As  $L_p$ -norm is directly computable and an approximation to information gain, we typically choose it to

measure the distance  $d(\beta)$ , which is also an approximation of the reduction in bits, i.e.,

$$F^{k+1} = \arg \max_{F^{(\beta)} \in \mathcal{B}/S_k} \frac{1}{2} |f^{(\beta)} - E_{p(\mathbf{I}; \Lambda_k, S_k)}[H^{(\beta)}]|_p. \quad (3.27)$$

For the following, we choose  $p = 1$ . To estimate  $f^{(\beta)}$  and  $E_{p(\mathbf{I}; \Lambda_k, S_k)}[H^{(\beta)}]$ , we applied  $F^{(\beta)}$  to the observed image  $\mathbf{I}^{obs}$  and the synthesized image  $\mathbf{I}^{syn}$  sampled from the  $p(\mathbf{I}; \Lambda_k, S_k)$  and substitute the histograms of the filtered images for  $f^{(\beta)}$  and  $E_{p(\mathbf{I}; \Lambda_k, S_k)}[H^{(\beta)}]$ , i.e.,

$$F^{k+1} = \arg \max_{F^{(\beta)} \in \mathcal{B}/S_k} \frac{1}{2} |H^{obs(\beta)} - H^{syn(\beta)}|. \quad (3.28)$$

The filter selection procedure stops when the  $d(\beta)$  for all filters  $F^{(\beta)}$  in the filter bank are smaller than a constant  $\epsilon$ . Due to fluctuation, various patches of the same observed texture image often have a certain amount of histogram variance, and we use such a variance for  $\epsilon$ .

Finally, we propose Algorithm 3 for filter selection.

Let  $\mathcal{B}$  be a bank of filters,  $S$  the set of selected filters,  $\mathbf{I}^{obs}$  the observed texture image, and  $\mathbf{I}^{syn}$  the synthesized texture image.

Initialize  $k = 0$ ,  $S \leftarrow \emptyset$ ,  $p(\mathbf{I}) \leftarrow$  uniform dist.  $\mathbf{I}^{syn} \leftarrow$  uniform noise.

For  $\alpha = 1, \dots, |\mathcal{B}|$  do

    Compute  $\mathbf{I}^{obs(\alpha)}$  by applying  $F^{(\alpha)}$  to  $\mathbf{I}^{obs}$ .

    Compute histogram  $H^{obs(\alpha)}$  of  $\mathbf{I}^{obs(\alpha)}$ .

Repeat

    For each  $F^{(\beta)} \in \mathcal{B}/S$  do

        Compute  $\mathbf{I}^{syn(\beta)}$  by applying  $F^{(\beta)}$  to  $\mathbf{I}^{syn}$

        Compute histogram  $H^{syn(\beta)}$  of  $\mathbf{I}^{syn(\beta)}$

$d(\beta) = \frac{1}{2} |H^{obs(\beta)} - H^{syn(\beta)}|_{\frac{1}{2}}$

    Choose  $F^{k+1}$  so that  $d(k+1) = \max\{d(\beta) : \forall F^{(\beta)} \in \mathcal{B}/S\}$

$S \leftarrow S \cup \{F^{k+1}\}$ ,  $k \leftarrow k + 1$ .

    Starting from  $p(\mathbf{I})$  and  $\mathbf{I}^{syn}$ , run algorithm 1 to compute new  $p^*(\mathbf{I})$  and  $\mathbf{I}^{syn*}$ .

$p(\mathbf{I}) \leftarrow p^*(\mathbf{I})$  and  $\mathbf{I}^{syn} \leftarrow \mathbf{I}^{syn*}$ .

Until  $d^{(\beta)} < \epsilon$ .

### Algorithm 3: Filter Selection

Fig. 3.19.a is the observed image of animal fur. We start from the uniform noise image in Fig. 3.19.b.

The first filter picked by the algorithm is a Laplacian of Gaussian filter  $LG(1.0)$  and its window size is  $5 \times 5$ . It has the largest error  $d(\beta) = 0.611$  among all the filters in the filters bank. Then we synthesize texture as shown in Fig. 3.19.c, which has almost the same histogram at the sub-band of this filter (the error  $d(\beta)$  drops to 0.035).

To provide a final insight into the pursuit process from a distribution point of view, we refer to Fig. 3.20. Note that this is only a 2D view of the very-high dimensional distribution space. First, selecting a filter gives a (high-dimensional) constrained set of probability distributions satisfying a given feature statistic. Then we

<sup>‡</sup>Since both histograms are normalized to have  $sum = 1$ , then  $error \in [0, 1]$ . We note this measure is robust with respect to the choice of the bin number  $L$  (e.g. we can take  $L = 16, 32, 64$ ), as well as the normalization of the filters.

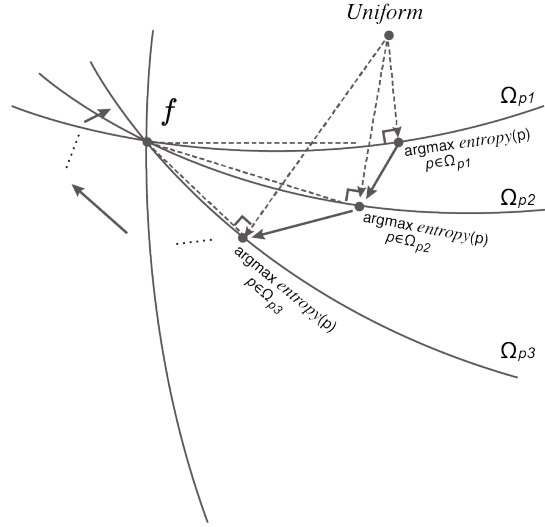


Figure 3.20: A graphical view of the FRAME learning algorithm in distribution space.

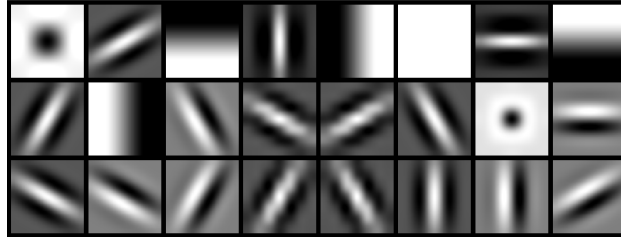


Figure 3.21: Note the selected filter after each filter selection. Each time, we add the filter that reduces  $KL(f, p(\mathbf{I}; \Lambda))$  the most.

try to look for a distribution in this constrained set that is the closest to the random uniform distribution (by Maximum Entropy Principle), as represented by the right angle. In fact, we are minimizing  $KL(p, p_0)$  where  $p \in \Omega_{p_1}$  and  $p_0$  is the starting distribution. This objective is equivalent to maximizing entropy if and only if  $p_0$  is random uniform. We leave this proof as an exercise for the reader. Second, we find the new filter that gives the largest reduction in  $KL(f, p)$  (by Minimum Entropy Principle), constraining the equivalence class of distributions further to a lower-dimensional set. It is important to note that  $\Omega_{p_2}$  has lower dimension than  $\Omega_{p_1}$  and every set after  $\Omega_{p_2}$  has lower dimension due to additional constraints. The true distribution  $f$  again also remains in each set, as it satisfies each additional feature statistic. In each new constrained set, we start again from the random uniform distribution and find a new closest distribution. The algorithm continues and we see from the solid line that our projected distribution "spirals" into the true data distribution as more constraints are put onto the distribution. Further details on information projection are presented in Chapter 9.

Fig. 3.21 is the filter we selected and added into the filter set, which we use it the model. As we can see in the figure, most of the filters we selected are Gabor filters and we also need some other filters to help detect the edges and corners.

Fig. 3.22 shows the curve of the weighted error per bin over the number of filters used for synthesis. As it is shown in the plot, after about 10 iterations, the weighted error is stable enough and can generate very

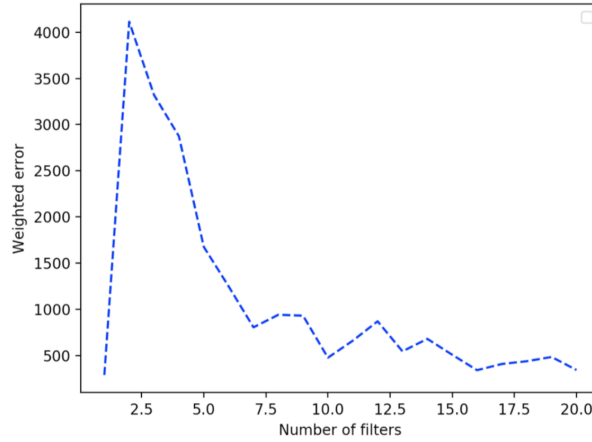


Figure 3.22: The weighted error over the number of filters used,

reasonable figures, which on the other hand perfectly match the result that given in Fig. 3.19.

### 3.5 The Texture Ensemble

The following Section presents theoretical work on texture theories and establishes connections between FRAME, Statistical Physics, and the Julesz Ensemble. This Section is highly technical and it is encouraged for interested readers to go through the original works [237] [236].

#### 3.5.1 Ensembles in Statistical Physics

We have developed the FRAME model based on the maximum entropy principle and in the following we will show that there is a simpler and more profound way to derive or justify the FRAME model. This insight again originates from Statistical Physics, specifically the equivalence of ensembles.

Statistical physics is the subject of studying macroscopic properties of a system involving a massive amount of elements (see Chandler, 1987 [23]). Fig. 5.5 illustrates three types of physical systems which are interesting to us.

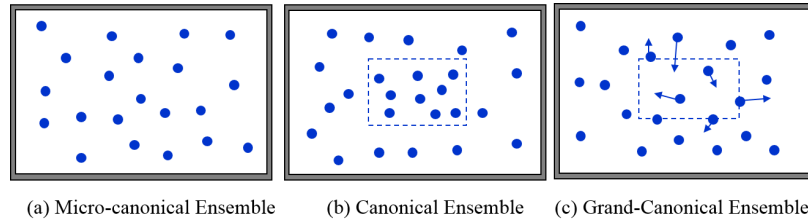


Figure 3.23: Three types of ensembles in statistical physics.

1. *Micro-canonical ensembles.* Fig. 5.5.a is an insulated system of  $N$  elements. The elements could be atoms, molecules, electrons in systems such as gas, ferro-magnetic material, fluid etc.  $N$  is big, say  $N = 10^{23}$  and is considered infinity. The system is decided by a configuration or state  $s = (\mathbf{x}^N, \mathbf{m}^N)$ , where  $\mathbf{x}^N$  describes the coordinates of the  $N$  elements and  $\mathbf{m}^N$  their momenta (Chandler, 1987). It is impractical to study the exact motions of  $6N$  vector  $s$ , and in fact, these microscopic states are less relevant,

and people are more interested in the macroscopic properties of the system as a whole, say the number of elements  $N$ , the total energy  $E(s)$ , and total volume  $V$ . Other derivative properties are temperature and pressure etc.

If we denote by  $\mathbf{h}(s) = (N, E, V)$  the macroscopic properties, at thermodynamic equilibrium all microscopic states that satisfy this property make up a *micro-canonical ensemble*:

$$\Omega_{mce}(\mathbf{h}_o) = \{s = (\mathbf{x}^N, \mathbf{m}^N) : \mathbf{h}(s) = \mathbf{h}_o = (N, V, E)\}.$$

$s$  is an *instance* and  $\mathbf{h}(s)$  is *summary* of the system state for practical purposes. Obviously  $\Omega_{mce}$  is a deterministic set or an equivalence class for all states that satisfy descriptive constraints  $\mathbf{h}(s) = \mathbf{h}_o$ .

An essential assumption in statistical physics is, as a first principle,

*“all microscopic states are equally likely at thermodynamic equilibrium.”*

This is simply a maximum entropy assumption. Let  $\Omega \ni s$  be the space of all possible states, then  $\Omega_{mce} \subset \Omega$  is associated with a uniform probability,

$$p_{\text{unif}}(s; \mathbf{h}_o) = \begin{cases} 1/|\Omega_{mce}(\mathbf{h}_o)| & \text{for } s \in \Omega_{mce}(\mathbf{h}_o), \\ 0 & \text{for } s \in \Omega/\Omega_{mce}(\mathbf{h}_o). \end{cases}$$

2. *Canonical ensembles.* The canonical ensemble refers to a small system (with fixed volume  $V_1$  and elements  $N_1$ ) embedded in a micro-canonical ensemble (see Fig. 5.5.b). The canonical ensemble can exchange energy with the rest of the system (called heat bath or reservoir). The system is relatively small, e.g.  $N_1 = 10^{10}$ , so that the bath can be considered a micro-canonical ensemble itself.

At thermodynamic equilibrium, the microscopic state  $s_1$  for the small system follows a Gibbs distribution,

$$p_{\text{Gib}}(s_1; \beta) = \frac{1}{Z} \exp\{-\beta E(s_1)\}.$$

The conclusion was stated as a general theorem by Gibbs (1902),

*“If a system of a great number of degrees of freedom is micro-canonically distributed in phase, any very small part of it may be regarded as canonically distributed.”*

In accordance with this theorem, the Gibbs model  $p_{\text{Gib}}$  is a conditional probability of the uniform model  $p_{\text{unif}}$ . This conclusion is extremely important because it bridges a deterministic set  $\Omega_{mce}$  with a descriptive model  $p_{\text{Gib}}$ . We consider this is an origin of probability in modeling visual patterns.

3. *Grand-Canonical ensembles.* When the small system with fixed volume  $V_1$  can also exchange elements with the bath as in liquid and gas materials, then it is called a grand-canonical ensemble (see Fig. 5.5.c). The grand-canonical ensemble follows a distribution,

$$p_{\text{gce}}(s_1; \beta_o, \beta) = \frac{1}{Z} \exp\{-\beta_o N_1 - \beta E(s_1)\},$$

where an extra parameter  $\beta_o$  controls the number of elements  $N_1$  in the ensemble. If we replace the energy of the physical system by the feature statistics of the texture image, then we will arrive at a mathematical model for textures.

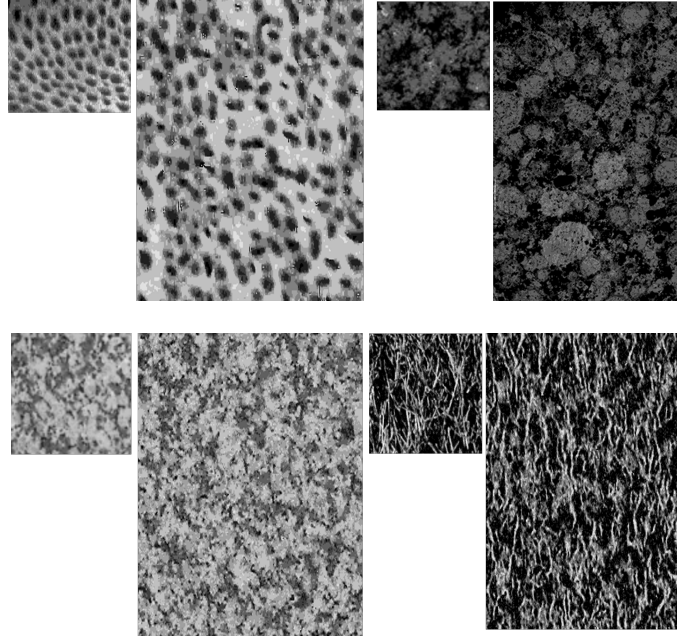


Figure 3.24: For each pair of texture images, the image on the left is the observed image, and the image on the right is the image randomly sampled from the Julesz texture ensemble.

### 3.5.2 Texture Ensemble

To study real world textures, one needs to characterize the dependency between pixels by extracting spatial features and calculating some statistics averaged over the image. One main theme of texture research is to seek the essential ingredients in terms of features and statistics  $\mathbf{h}(\mathbf{I})$ , which are the bases for human texture perception. From now on, we use the bold font  $\mathbf{h}$  to denote statistics of image features.

In the literature, the search for  $\mathbf{h}$  has converged to marginal histograms of Gabor filter responses. We believe that some bins of joint statistics may also be important as long as they can be reliably estimated from finite observations.

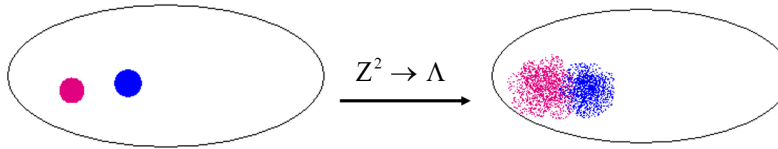


Figure 3.25: Julesz ensembles of different textures are disjoint in the space of infinite images. They will overlap in the space of image patches of finite size.

Given  $K$  Gabor filters as feature detectors  $\{F^{(1)}, \dots, F^{(K)}\}$ , we convolve the filters with image  $\mathbf{I}$  to obtain the sub-band filtered images  $\{\mathbf{I}^{(1)}, \dots, \mathbf{I}^{(K)}\}$ , where  $\mathbf{I}^{(k)} = F^{(k)} * \mathbf{I}$ . Let  $h^{(k)}$  be the normalized intensity histogram of  $\mathbf{I}^{(k)}$ ; then the feature statistics  $\mathbf{h}$  collects the normalized histograms of these  $K$  sub-band images,

$$\mathbf{h}(\mathbf{I}) = (h^{(1)}(\mathbf{I}), \dots, h^{(K)}(\mathbf{I})).$$

We use  $\mathbf{H}(\mathbf{I}) = (H^{(1)}(\mathbf{I}), \dots, H^{(K)}(\mathbf{I}))$  to denote the unnormalized histograms. We assume that boundary conditions are properly handled (e.g., periodic boundary condition). It should be noted that the conclu-

sions here hold as long as  $\mathbf{h}(\mathbf{I})$  can be expressed as spatial averages of local image features. The marginal histograms of Gabor filter responses are only special cases.

Given statistics  $\mathbf{h}(\mathbf{I})$ , one can partition the image space  $\Omega_D$  into equivalence classes  $\Omega_D(\mathbf{h}) = \{\mathbf{I} : \mathbf{h}(\mathbf{I}) = \mathbf{h}\}$ . For finite  $D$ , the exact constraint  $\mathbf{h}(\mathbf{I}) = \mathbf{h}$  may not be satisfied, so we relax this constraint, and replace  $\Omega_D(\mathbf{h})$  by

$$\Omega_D(\mathcal{H}) = \{\mathbf{I} : \mathbf{h}(\mathbf{I}) \in \mathcal{H}\}$$

with  $\mathcal{H}$  being a small set around  $\mathbf{h}$ . Then we can define the uniform counting measure or the uniform probability distribution on  $\Omega_D(\mathcal{H})$  as

$$q(\mathbf{I}; \mathcal{H}) = \begin{cases} 1/|\Omega_D(\mathcal{H})|, & \text{if } \mathbf{I} \in \Omega_D(\mathcal{H}), \\ 0, & \text{otherwise,} \end{cases} \quad (3.29)$$

where  $|\Omega_D(\mathcal{H})|$  is the volume of or the number of images in  $\Omega_D(\mathcal{H})$ . Now we can define the Julesz ensemble as follows.

**Definition.** Given a set of feature statistics  $\mathbf{h}(\mathbf{I}) = (h^{(1)}(\mathbf{I}), \dots, h^{(K)}(\mathbf{I}))$ , a Julesz ensemble of type  $\mathbf{h}$  is a limit of  $q(\mathbf{I}; \mathcal{H})$  as  $D \rightarrow \mathbb{Z}^2$  and  $\mathcal{H} \rightarrow \mathbf{h}$  with some boundary condition.<sup>§</sup>

As in the simple iid example, the Julesz ensemble is defined mathematically as the limit of a uniform counting measure. It is always helpful to imagine the Julesz ensemble of type  $\mathbf{h}$  as the image set  $\Omega_D(\mathbf{h})$  on a large  $D$ . Also, in the later calculation, we shall often ignore the minor complication that constraint  $\mathbf{h}(\mathbf{I}) = \mathbf{h}$  may not be exactly satisfied.

Then, we are ready to give a mathematical definition for textures.

**Definition.** A texture pattern is a Julesz ensemble defined by a type  $\mathbf{h}$  of the feature statistics  $\mathbf{h}(\mathbf{I})$ .

### 3.5.3 Type Theory and Entropy Rate Functions

#### A Simple Independent Model

In this section, we introduce basic concepts, such as type, ensemble, entropy function, typical images, and equivalence of ensembles, using a simple image model where the pixel intensities are independently and identically distributed (iid).

Let  $\mathbf{I}$  be an image defined on a finite lattice  $D \subset \mathbb{Z}^2$ , and the intensity at pixel  $v \in D$  is denoted by  $\mathbf{I}(v) \in \mathcal{G} = \{1, 2, \dots, g\}$ . Thus  $\Omega_D = \mathcal{G}^{|D|}$  is the space of images on  $D$ , with  $|D|$  being the number of pixels in  $D$ .

1) *The FRAME model for iid images.* We consider a simple image model where pixel intensities are independently and identically distributed according to a probability  $p = (p_1, \dots, p_g)$  where  $\sum_i p_i = 1$ . The distribution of  $\mathbf{I}$  can be written as a FRAME model

$$p(\mathbf{I}; \beta) = \prod_{v \in D} p_{\mathbf{I}(v)} = \prod_{i=1}^g p_i^{H_i(\mathbf{I})} = \exp\{\langle \log p, H(\mathbf{I}) \rangle\} = \exp\{\langle \beta, H(\mathbf{I}) \rangle\}, \quad (3.30)$$

where  $H(\mathbf{I}) = (H_1(\mathbf{I}), \dots, H_g(\mathbf{I}))$  is the unnormalized intensity histogram of  $\mathbf{I}$ , i.e.,  $H_i$  is the number of pixels whose intensities equal to  $i$ .  $\beta = (\log p_1, \dots, \log p_g)$  is the parameter of  $p(\mathbf{I}; \beta)$  – a special case of the FRAME model.

2) *Type.* Let  $h(\mathbf{I}) = H(\mathbf{I})/|D|$  be the normalized intensity histogram. We call  $h(\mathbf{I})$  the *type* of image  $\mathbf{I}$ .

---

<sup>§</sup>We assume  $D \rightarrow \mathbb{Z}^2$  in the sense of van Hove, i.e., the ratio between the boundary and the size of  $D$  goes to 0.

3) *Equivalent class.* Let  $\Omega_D(h)$  be the set of images with  $h(\mathbf{I}) = h$ <sup>‡</sup>, i.e.,  $\Omega_D(h) = \{\mathbf{I} : h(\mathbf{I}) = h\}$ . Then the image space is partitioned into equivalence classes

$$\Omega_D = \cup_h \Omega_D(h).$$

As shown in Fig. 3.26, each equivalence class  $\Omega_D(h)$  is mapped into one type  $h$  on a *simplex* – a plane defined by  $h_1 + \dots + h_g = 1$  and  $h_i \geq 0, \forall i$  in an  $g$ -dimensional space.

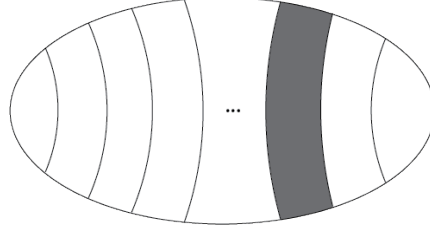


Figure 3.26: The partition of the image space into equivalence classes, and each class corresponds to an  $h$  on the probability simplex.

4) *The Julesz ensemble for iid images.* The hard constraint in defining the equivalence class  $\Omega_D(h)$  makes sense only in the limit as  $D \rightarrow \mathbb{Z}^2$ , where statistical fluctuations vanish. Therefore, we may attempt to define the Julesz ensemble as the limit of  $\Omega_D(h)$  as  $D \rightarrow \mathbb{Z}^2$ , or even more directly, as the set of images  $\mathbf{I}$  defined on  $\mathbb{Z}^2$  with  $h(\mathbf{I}) = h$ .

Unfortunately, the above “definitions” are not mathematically well-defined. Instead, we need to define the Julesz ensemble in a slightly indirect way. First, we associate with each equivalence class  $\Omega_D(h)$  a probability distribution  $q(\mathbf{I}; h)$ , which is uniform over  $\Omega_D(h)$  and vanishes outside. Then, the *Julesz ensemble* of type  $h$  is defined to be the limit of  $q(\mathbf{I}; h)$  as  $D \rightarrow \mathbb{Z}^2$ .

For finite  $D$ , the equivalence class  $\Omega_D(h)$  may be empty because  $|D|h$  may not be integers. Thus, to be more rigorous, we should replace  $h$  by a small set  $\mathcal{H}$  around  $h$ , and let  $\mathcal{H}$  go to  $h$  as  $D \rightarrow \infty$ . For simplicity, however, we shall neglect this minor complication and simply treat  $|D|h$  as integers.

The uniform distribution  $q(\mathbf{I}; h)$  only serves as a counting measure of the equivalence class  $\Omega_D(h)$ , i.e., all the images in  $\Omega_D(h)$  are counted equally. Therefore, any probability statement under the uniform distribution  $q(\mathbf{I}; h)$  is equivalent to a frequency statement of images in  $\Omega_D(h)$ . For example, the probability that image  $\mathbf{I}$  has a certain property under  $q(\mathbf{I}; h)$  is actually the frequency or the proportion of images in  $\Omega_D(h)$  that have this property. The limit of  $q(\mathbf{I}; h)$  thus essentially defines a counting measure of the set of infinitely large images (defined on  $\mathbb{Z}^2$ ) with histogram  $h$ . With a little abuse of language, we sometimes also call the equivalence class  $\Omega_D(h)$  defined on a large lattice  $D$  a Julesz ensemble, and it is always helpful to imagine a Julesz ensemble as such an equivalence class if the reader finds the limit of probability measures too abstract.

5) *Entropy function.* We are interested in computing the volume of the Julesz ensemble  $\Omega_D(h)$ , i.e., the number of images in  $\Omega_D(h)$ . We denote this volume by  $|\Omega_D(h)|$ . Clearly

$$|\Omega_D(h)| = \frac{|D|!}{\prod_{i=1}^g (h_i |D|)!}.$$

Using the Stirling formula, it can be easily shown that

---

<sup>‡</sup>We hope that the notation  $h(\mathbf{I}) = h$  will not confuse the reader. The  $h$  on the left is a function of  $\mathbf{I}$  for extracting statistics, while the  $h$  on the right is a specific value of the statistics.



$$\begin{aligned}
\lim_{D \rightarrow \mathbb{Z}^2} \frac{1}{|D|} \log |\Omega_D(h)| &= \lim_{D \rightarrow \mathbb{Z}^2} \frac{1}{|D|} \log \frac{|D|!}{\prod_{i=1}^g (h_i |D|)!} \\
&= - \sum_{i=1}^g h_i \log h_i = \text{entropy}(h).
\end{aligned}$$

Thus for large enough lattice, the volume of  $\Omega_D(h)$  is said to be in the order of  $\text{entropy}(h)$ , i.e.,

$$|\Omega_D(h)| \sim \exp^{|D| \text{entropy}(h)}.$$

For notational simplicity, we denote the entropy function by  $s(h) = \text{entropy}(h)$ .

6) *The probability rate function.* Now we are ready to compute the total probability mass that  $p(\mathbf{I}; \beta)$  assigns to an equivalence class  $\Omega_D(h)$ . We denote this probability by  $p(\Omega_D(h); \beta)$ . Because images in  $\Omega_D(h)$  all receive equal probabilities, it can be shown that

$$\begin{aligned}
\lim_{D \rightarrow \mathbb{Z}^2} \frac{1}{|D|} \log p(\Omega_D(h); \beta) &= \lim_{D \rightarrow \mathbb{Z}^2} \frac{1}{|D|} \log \{ |\Omega_D(h)| \prod_{i=1}^g p_i^{|D| h_i} \} \\
&= - \sum_{i=1}^g h_i \log \frac{h_i}{p_i} = -\text{KL}(h||p),
\end{aligned}$$

where  $\text{KL}(h||p)$  denotes the Kullback-Leibler distance from  $h$  to  $p$ .  $\text{KL}(h||p) \geq 0$  for all  $h$  and  $p$ , with equality holding when  $h = p$ .

Thus, on a large enough lattice, the total probability mass of an equivalence class  $\Omega_D(h)$  is said to be on the order of  $-\text{KL}(h||p)$ ,

$$p(\Omega_D(h); \beta) \sim \exp^{-|D| \text{KL}(h||p)} \quad (3.31)$$

$-\text{KL}(h||p)$  is the probability rate function, and is denoted by  $s_\beta(h) = -\text{KL}(h||p)$ .

7) *Typical vs. most likely images.* Suppose among  $p_1, \dots, p_g, p_m$  is the largest probability. Consider one extreme type  $h$ , with  $h_m = 1$ , and  $h_i = 0, \forall i \neq m$ . Then the image in this  $\Omega_D(h)$  is the most likely image under model  $p(\mathbf{I}; \beta)$ , i.e., it receives the highest probability. However,  $\Omega_D(h)$  has only one constant image, and the probability that  $p(\mathbf{I}; \beta)$  assigns to this  $\Omega_D(h)$  is essentially zero for large lattice. Now consider the equivalence class  $\Omega_D(h = p)$ . Each image in  $\Omega_D(h = p)$  receives less probability than the most likely image, but overall, the total probability received by the whole  $\Omega_D(h = p)$  is essentially 1 for large lattice. That is, if we sample from the FRAME model  $p(\mathbf{I}; \beta)$ , we will almost always get an image  $\mathbf{I}$  from  $\Omega_D(h = p)$ . Such images are called the *typical images* of the FRAME model.

Having introduced the basic concepts, we now explain the basic ideas of ensemble equivalence in the next two subsections by going both directions from one to the other.

### From a FRAME model to a Julesz ensemble on Infinite Lattice

We study the limit of the FRAME model  $p(\mathbf{I}; \beta)$  as  $D \rightarrow \mathbb{Z}^2$ . A simple fact will be repeatedly used in this section. To see this fact, let's consider the following example. Suppose we have two terms, one is  $e^{5n}$ , and the other is  $e^{3n}$ . Consider their sum  $e^{5n} + e^{3n}$ . As  $n \rightarrow \infty$ , the sum  $e^{5n} + e^{3n}$  is dominated by  $e^{5n}$ , and the order of this sum is still 5, i.e.,  $\lim_{n \rightarrow \infty} \frac{1}{n} \log(e^{5n} + e^{3n}) = 5$ . This means that for the sum of many terms, the term with the largest exponential order dominates the sum, and the order of the sum is the largest order among the individual terms.

Suppose  $\mathcal{H}$  is a set of types and

$$\Omega_D(\mathcal{H}) = \{\mathbf{I}; h(\mathbf{I}) \in \mathcal{H}\}$$

is the set of all images  $\mathbf{I}$  whose type is within  $\mathcal{H}$ . Then from Equation 3.31, we have

$$\begin{aligned} p(\mathbf{I} \in \Omega_D(\mathcal{H}); \beta) &= \int_{\mathcal{H}} p(\Omega_D(h); \beta) dh \\ &\sim \int_{\mathcal{H}} \exp^{-|D| \text{KL}(h||p)} dh \sim \exp^{-|D| \text{KL}(h^*||p)}. \end{aligned}$$

where  $h^*$  is the type in  $\mathcal{H}$  which has minimum Kullback-Leibler divergence to  $p$ ,

$$h^* = \arg \min_{h \in \mathcal{H}} \text{KL}(h||p).$$

That is, the total probability for  $\Omega_D(\mathcal{H})$  has an exponential order  $\text{KL}(h^*||p)$  and is dominated by the heaviest type  $h^*$ . In a special case  $\Omega_D(\mathcal{H}) = \Omega_D$ , i.e., the whole image space, we have  $h^* = p$  and  $\text{KL}(h^*||p) = 0$ .

Clearly, as  $|D| \rightarrow \infty$ , the FRAME model quickly concentrates its probability mass on  $\Omega_D(h = p)$ , and assigns equal probabilities to images in  $\Omega_D(h = p)$ . For other  $h \neq p$ , the probabilities will decrease to 0 at an exponential rate. Thus, the FRAME model becomes a Julesz ensemble.

### From a Julesz Ensemble to a FRAME model on Finite Lattice

In this section, we tighten up the notation a little bit. We use  $\mathbf{I}_D$  to denote the image defined on lattice  $D$ , and we use  $\mathbf{I}_{D_0}$  to denote the image patch defined on  $D_0 \subset D$ . For a fixed type  $h$  of feature statistics, consider the uniform distribution  $q(\mathbf{I}; h)$  on  $\Omega_D(h)$ . Under  $q(\mathbf{I}; h)$ , the distribution of  $\mathbf{I}_{D_0}$ , denoted by  $q(\mathbf{I}_{D_0}; h)$ , is well defined.<sup>||</sup> We shall show that if we fix  $D_0$  and let  $D \rightarrow \mathbb{Z}^2$ , then  $q(\mathbf{I}_{D_0}; h)$  goes to the FRAME model (see Equation 3.30) with  $p = h$ .

First, the number of images in  $\Omega_D(h)$  is

$$|\Omega_D(h)| = \frac{|D|!}{\prod_{i=1}^m (h_i |D|)!}.$$

Then, let's fix  $\mathbf{I}_{D_0}$  and calculate the number of images in  $\Omega_D(h)$  whose image value (i.e., intensities) on  $D_0$  is  $\mathbf{I}_{D_0}$ . Clearly, for every such image, its image value on the rest of the lattice  $D/D_0$ , i.e.,  $\mathbf{I}_{D/D_0}$  must satisfy

$$H(\mathbf{I}_{D/D_0}) = h|D| - H(\mathbf{I}_{D_0}),$$

where  $H(\mathbf{I}_{D_0}) = |D_0| h(\mathbf{I}_{D_0})$  is the unnormalized histogram of  $\mathbf{I}_{D_0}$ . Therefore

$$\mathbf{I}_{D/D_0} \in \Omega_{D/D_0} \left( \frac{h|D| - H(\mathbf{I}_{D_0})}{|D/D_0|} \right).$$

So the number of such images is  $|\Omega_{D/D_0}((h|D| - H(\mathbf{I}_{D_0}))/|D/D_0|)|$ . Thus,

$$q(\mathbf{I}_{D_0}; h) = \frac{|\Omega_{D/D_0}(\frac{h|D| - H(\mathbf{I}_{D_0})}{|D/D_0|})|}{|\Omega_D(h)|}$$

---

<sup>||</sup>In the iid case,  $q(\mathbf{I}_{D_0}; h)$  is both the marginal distribution and the conditional distribution of  $q(\mathbf{I}; h)$ , while in random fields, we only consider the conditional distribution.

$$\begin{aligned}
&= \frac{(|D| - |D_0|)! / \prod_{i=1}^g (h_i |D| - H_i(\mathbf{I}_{D_0}))!}{|D|! / \prod_{i=1}^g (h_i |D|)!} \\
&= \frac{\prod_{i=1}^g (h_i |D|)(h_i |D| - 1) \dots (h_i |D| - H_i(\mathbf{I}_{D_0}) + 1)}{|D|(|D| - 1) \dots (|D| - |D_0| + 1)} \\
&= \frac{\prod_{i=1}^g h_i (h_i - 1/|D|) \dots (h_i - (H_i(\mathbf{I}_{D_0}) - 1)/|D|)}{(1 - 1/|D|) \dots (1 - (|D_0| - 1)/|D|)} \\
&\rightarrow \prod_{i=1}^g h_i^{H_i(\mathbf{I}_{D_0})} \text{ as } |D| \rightarrow \infty.
\end{aligned}$$

Therefore, the distribution of  $\mathbf{I}_{D_0}$  is the FRAME model (see Equation 3.30) with  $p = h$  under the Julesz ensemble defined by  $h$ .

The above calculation can be easily interpreted in a non-probabilistic way. That is,  $q(\mathbf{I}_{D_0}; h)$  is the frequency or the proportion of images in  $\Omega_D(h)$  (on large  $D$ ) whose patches on  $D_0$  are  $\mathbf{I}_{D_0}$ . In other words, if we look at all the images in the Julesz ensemble through  $D_0$ , then we will find a collection of images on  $D_0$ , and the distribution of this collection is described by the FRAME model. The reason that the FRAME model assigns probabilities to all images on  $D_0$  is also quite clear. Under the hard constraint on  $h(\mathbf{I}_D)$ ,  $h(\mathbf{I}_{D_0})$  can still take any possible values.

### 3.5.4 Equivalence of FRAME and Julesz Ensemble

In this section, we show the equivalence between the Julesz ensembles and the FRAME models, using the fundamental principle of equivalence of ensembles in statistical mechanics.

#### From the Julesz Ensemble to the FRAME Model

In this subsection, we derive the local Markov property of the Julesz ensemble, which is globally defined by type  $\mathbf{h}$ . This derivation is adapted from a traditional argument in statistical physics (e.g., Laudu, 1959). It is not as rigorous as modern treatments, but it is much more revealing.

Suppose the feature statistics are  $\mathbf{h}(\mathbf{I})$  where  $\mathbf{I}$  is defined on  $D$ . For a fixed value of feature statistics  $\mathbf{h}$ , consider the image set  $\Omega_D(\mathbf{h}) = \{\mathbf{I} : \mathbf{h}(\mathbf{I}) = \mathbf{h}\}$ , and the associated uniform distribution  $q(\mathbf{I}; \mathbf{h})$ . First, we fix  $D_1 \subset D$ , and then fix  $D_0 \subset D_1$ . We are interested in the conditional distribution of the local patch  $\mathbf{I}_{D_0}$  given its local environment  $\mathbf{I}_{D_1/D_0}$  under the model  $q(\mathbf{I}; \mathbf{h})$  as  $D \rightarrow \mathbb{Z}^2$ . We assume that  $D_0$  is sufficiently smaller than  $D_1$  so that the neighborhood of  $D_0$ ,  $\partial D_0$ , is contained in  $D_1$ .

Let  $\mathbf{H}_0 = \mathbf{H}(\mathbf{I}_{D_0} | \mathbf{I}_{\partial D_0})$  be the unnormalized statistics computed for  $\mathbf{I}_{D_0}$  where filtering takes place within  $D_0 \cup \partial D_0$ . Let  $\mathbf{H}_{01}$  be the statistics computed by filtering inside the fixed environment  $D_1/D_0$ . Let  $D_{-1} = D/D_1$  be the big patch outside of  $D_1$ . Then the statistics computed for  $D_{-1}$  is  $\mathbf{h}|D| - \mathbf{H}_0 - \mathbf{H}_{01}$ . Let  $\mathbf{h}_- = (\mathbf{h}|D| - \mathbf{H}_{01})/|D_{-1}|$ , then the normalized statistics for  $D_{-1}$  is  $\mathbf{h}_- - \mathbf{H}_0/|D_{-1}|$ .

For a certain  $\mathbf{I}_{D_0}$ , the number of images in  $\Omega_D(\mathbf{h})$  with such a patch  $\mathbf{I}_{D_0}$  and its local environment  $\mathbf{I}_{D_1/D_0}$  is  $|\Omega_{D_{-1}}(\mathbf{h}_- - \mathbf{H}_0/|D_{-1}|)|$ . Therefore the conditional probability, as a function of  $\mathbf{I}_{D_0}$ , is

$$q(\mathbf{I}_{D_0} | \mathbf{I}_{D_1/D_0}, \mathbf{h}) \propto |\Omega_{D_{-1}}(\mathbf{h}_- - \frac{\mathbf{H}_0}{|D_{-1}|})|.$$

Unlike the simple iid case, however, the above volume cannot be computed analytically. However, the volume  $|\Omega_D(\mathbf{h})|$  still shares the same asymptotic behavior as that of the simple iid example, namely,

$$\lim_{D \rightarrow \mathbb{Z}^2} \frac{1}{|D|} \log |\Omega_D(\mathbf{h})| \rightarrow s(\mathbf{h}),$$

where  $s(\mathbf{h})$  is a concave entropy function of  $\mathbf{h}$ .

Like the simple iid case, in the above derivation, we ignore the minor technical complication that  $\Omega_D(\mathbf{h})$  may be empty because the exact constraint may not be satisfied on finite lattice. A more careful treatment is to replace  $\mathbf{h}$  by a small set  $\mathcal{H}$  around  $\mathbf{h}$ , and let  $\mathcal{H} \rightarrow \mathbf{h}$  as  $D \rightarrow Z^2$ . Let  $\Omega_D(\mathcal{H}) = \{\mathbf{I} : \mathbf{h}(\mathbf{I}) \in \mathcal{H}\}$ , then we have the following

**Proposition 3.** *The limit*

$$\lim_{D \rightarrow Z^2} \frac{1}{|D|} \log \Omega_D(\mathcal{H}) = s(\mathcal{H})$$

*exists. Let  $s(\mathbf{h}) = \lim_{\mathcal{H} \rightarrow \bar{\mathcal{H}}} s(\mathcal{H})$ , then  $s(\mathbf{h})$  is concave, and  $s(\mathcal{H}) = \sup_{\bar{\mathcal{H}} \in \mathcal{H}} s(\bar{\mathcal{H}})$ .*

See Lanford (1973) for a detailed analysis of the above result. The  $s(\mathbf{h})$  is a measure of the volume of the Julesz ensemble of type  $\mathbf{h}$ . It defines the randomness of the texture appearance of type  $\mathbf{h}$ .

With such an estimate, we are ready to compute the conditional probability. Note that the conditional distribution,  $q(\mathbf{I}_{D_0} \mid \mathbf{I}_{D_1/D_0}, \mathbf{h})$ , as a function of  $\mathbf{I}_{D_0}$ , is decided only by  $\mathbf{H}_0$ , which is the sufficient statistics. Therefore, we only need to trace  $\mathbf{H}_0$  while leaving other terms as constants. For large  $D$ , a Taylor expansion at  $\mathbf{h}_-$  gives

$$\begin{aligned} \log q(\mathbf{I}_{D_0} \mid \mathbf{I}_{D_1/D_0}, \mathbf{h}) &= \text{constant} + \log |\Omega_{D-1}(\mathbf{h}_- - \frac{\mathbf{H}_0}{|D_{-1}|})| \\ &= \text{constant} + |D_{-1}| s(\mathbf{h}_- - \frac{\mathbf{H}_0}{|D_{-1}|}) \\ &= \text{constant} - \langle s'(\mathbf{h}_-), \mathbf{H}_0 \rangle + o(\frac{1}{|D|}). \end{aligned}$$

Assuming the entropy function  $s$  has continuous derivative at  $\mathbf{h}$ , and let  $\beta = s'(\mathbf{h})$ , then, as  $D \rightarrow Z^2$ ,  $\mathbf{h}_- \rightarrow \mathbf{h}$ , and  $s'(\mathbf{h}_-) \rightarrow \beta$ . Therefore,

$$\begin{aligned} \log q(\mathbf{I}_{D_0} \mid \mathbf{I}_{D_1/D_0}, \mathbf{h}) &\rightarrow \text{const} - \langle s'(\mathbf{h}), \mathbf{H}_0 \rangle \\ &= \text{const} - \langle \beta, \mathbf{H}_0 \rangle, \end{aligned}$$

so

$$q(\mathbf{I}_{D_0} \mid \mathbf{I}_{D_1/D_0}, \mathbf{h}) \rightarrow \frac{1}{Z_{D_0}(\beta)} \exp\{-\langle \beta, \mathbf{H}(\mathbf{I}_{D_0} \mid \mathbf{I}_{D_1/D_0}) \rangle\},$$

which is exactly the Markov property specified by the FRAME model. This derivation shows that local computation using the FRAME model is justified under the Julesz ensemble. It also reveals an important relationship, i.e., the parameter  $\beta$  can be identified as the derivative of the entropy function  $s(\mathbf{h})$ ,  $\beta = s'(\mathbf{h})$ .

As in the simple iid case, this result can be interpreted in a non-probabilistic way in terms of frequencies.

### From the FRAME Model to the Julesz Ensemble

In this subsection, we study the statistical properties of the FRAME models as  $D \rightarrow Z^2$ .

Consider the FRAME model

$$p(\mathbf{I}; \beta) = \frac{1}{Z(\beta)} \exp\{-|D| \langle \beta, \mathbf{h}(\mathbf{I}) \rangle\},$$

which assigns equal probabilities to the  $|\Omega_D(\mathbf{h})|$  images in  $\Omega_D(\mathbf{h})$ . The probability that  $p(\mathbf{I}; \beta)$  assigns to  $\Omega_D(\mathbf{h})$  is

$$p(\Omega_D(\mathbf{h}); \beta) = \frac{1}{Z_D(\beta)} \exp\{-|D| \langle \beta, \mathbf{h} \rangle\} |\Omega_D(\mathbf{h})|.$$

The asymptotic behavior of this probability is

$$\begin{aligned} s_\beta(\mathbf{h}) &= \lim_{D \rightarrow \mathbb{Z}^2} \frac{1}{|D|} \log p(\Omega_D(\mathbf{h}); \beta) \\ &= -\langle \beta, \mathbf{h} \rangle + s(\mathbf{h}) - \lim_{D \rightarrow \mathbb{Z}^2} \frac{1}{|D|} \log Z_D(\beta). \end{aligned}$$

For the last term, we have

**Proposition 4.** *The limit*

$$\rho(\beta) = \lim_{D \rightarrow \mathbb{Z}^2} \frac{1}{|D|} \log Z_D(\beta)$$

*exists and is independent of the boundary condition.  $\rho(\beta)$  is convex.*

See Griffiths and Ruelle (1971) for a proof. Therefore, we have

**Proposition 5.** *The probability rate function  $s_\beta(\mathbf{h})$  of the FRAME model  $p(\mathbf{I}; \beta)$  is*

$$s_\beta(\mathbf{h}) = \lim_{D \rightarrow \mathbb{Z}^2} \frac{1}{|D|} \log p(\Omega_D(\mathbf{h}); \beta) = s(\mathbf{h}) - \langle \beta, \mathbf{h} \rangle - \rho(\beta).$$

Therefore, the probability that the FRAME model puts on  $\Omega_D(\mathbf{h})$  behaves like  $\exp\{|D|s_\beta(\mathbf{h})\}$ , and clearly  $s_\beta(\mathbf{h}) \leq 0$  for any  $\mathbf{h}$  and any  $\beta$  (otherwise, the probability will go unbounded). This  $s_\beta(\mathbf{h})$  can be identified with  $-\text{KL}(h||p)$  in the simple iid case.

The probability rate function  $s_\beta(\mathbf{h})$  tells us that  $p(\mathbf{I}; \beta)$  will eventually concentrate on the  $\mathbf{h}_*$  that maximizes  $s_\beta(\mathbf{h}) = s(\mathbf{h}) - \langle \beta, \mathbf{h} \rangle - \rho(\beta)$ , or  $s(\mathbf{h}) - \langle \beta, \mathbf{h} \rangle$ . So we have the following

**Theorem 2.** *If there is a unique  $\mathbf{h}_*$  where  $s_\beta(\mathbf{h})$  achieves its maximum 0, then  $p(\mathbf{I}; \beta)$  eventually concentrates on  $\mathbf{h}_*$  as  $D \rightarrow \mathbb{Z}^2$ . Therefore the FRAME model  $p(\mathbf{I}; \beta)$  goes to a Julesz ensemble defined by  $\mathbf{h}_*$ . If  $s(\mathbf{h})$  is differentiable at  $\mathbf{h}_*$ , then  $s'(\mathbf{h}_*) = \beta$ .*

The uniqueness of  $\mathbf{h}_*$  holds under the condition that there is no phase transition at  $\beta$ .

### 3.6 Deriving Partial Differential Equations from the MRF and FRAME Models

In the previous sections, we have motivated feature extraction using filters and matching marginal distributions of observed data. This design of features has led to FRAME, a unified view of both clique-based and filter-based methods in texture modeling. Using Gibbs distribution as the probability distribution for textures, we further learn its potential function with dynamics

$$\frac{d\lambda^{(\alpha)}}{dt} = E_{p(\mathbf{I}; \Lambda_K, S_K)}[H^{(\alpha)}] - H^{obs(\alpha)}.$$

as in Equation 3.21.

Computing  $E_{p(\mathbf{I}; \Lambda_K, S_K)}[H^{(\alpha)}]$  is difficult and it involves sampling synthetic images using the current model distribution. A class of methods for image synthesis involves non-linear Partial Differential Equations (PDEs) in the form of

$$\frac{d\mathbf{I}}{dt} = T(\mathbf{I})$$

where  $T$  is a function of current  $\mathbf{I}$ . And in this section, we connect Gibbs distribution to PDE paradigms for texture formation and derive a common framework under which many previous PDE methods can be similarly derived.

We first introduce historical methods adopting PDEs for image processing inspired by physics and chemistry, then we introduce our discovery of reaction-diffusion functions as Gibbs potential functions, leading to Gibbs Reaction and Diffusion Equation (GRADE) as a family of PDEs for texture patterns formation.

### 3.6.1 Turing Diffusion-Reaction

A set of non-linear PDEs was first studied in [216] for modeling the formation of animal coat patterns by the diffusion and reaction of chemicals, which Turing called the “morphogens.” These equations were further explored by Murray in theoretical biology [163]. For example, let  $A(x, y, t)$  and  $B(x, y, t)$  be the concentrations of two morphogens at location  $(x, y)$  and time  $t$ , the typical reaction-diffusion equations are

$$\begin{cases} \frac{\partial A}{\partial t} = D_a \Delta A + R_a(A, B) \\ \frac{\partial B}{\partial t} = D_b \Delta B + R_b(A, B) \end{cases} \quad (3.32)$$

where  $D_a, D_b$  are constraints,  $\Delta = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$  is the Laplace operator, and  $R_a(A, B), R_b(A, B)$  are non-linear functions for the reaction of chemicals, e.g.,  $R_a(A, B) = A * B - A - 12$  and  $R_b(A, B) = 16 - A * B$ .

The morphogen theory provides a way for synthesizing some texture patterns. In the texture synthesis experiments, chemical concentrations are replaced by various colors, and the equations run for a finite steps with free boundary condition starting with some initial patterns. In some cases, both the initial patterns and the running processes have to be controlled manually in order to generate realistic textures. Two canonical textures synthesized by the Turing reaction-diffusion equation are the leopard blobs and zebra stripes.

In the reaction-diffusion equation above, the reaction terms are responsible for pattern formation, however they also make the equations unstable or unbounded. Even for a small system, the existence and stability problems for these PDEs are intractable (Grindrod 1996) [78]. In fact, we believe that running any non-linear PDEs for a finite steps will render some patterns, but it is unknown how to design a set of PDEs for a given texture pattern.

### 3.6.2 Heat Diffusion

As introduced in Section 3.2.3, generating image according to the heat diffusion equation

$$\frac{d\mathbf{I}}{dt} = \Delta \mathbf{I}(x, y)$$

is equivalent to minimizing potential energy of GMRF and

$$\frac{d\mathbf{I}}{dt} = - \frac{\delta U(\mathbf{I}(x, y, t))}{\delta \mathbf{I}}$$

where  $U(\mathbf{I}(x, y)) = \beta \int_x \int_y (\nabla_x \mathbf{I}(x, y))^2 + (\nabla_y \mathbf{I}(x, y))^2 dy dx$ .

In the following we show that Anisotropic diffusion can also be written in a similar form of minimizing a potential energy.

### 3.6.3 Anisotropic Diffusion

Perona and Malik introduced a family of anisotropic diffusion equations for generating image scale space  $\mathbf{I}(x, y, t)$  [178]. Similar to heat diffusion equation, their equation also simulates the “heat” diffusion process,

$$\frac{d\mathbf{I}}{dt} = \text{div}(c(x, y, t)\nabla\mathbf{I}), \quad \mathbf{I}(x, y, 0) = \mathbf{I}_0 \quad (3.33)$$

where  $\nabla\mathbf{I} = (\frac{\partial\mathbf{I}}{\partial x}, \frac{\partial\mathbf{I}}{\partial y})$  is the intensity gradient and  $\text{div}(\vec{V}) = \nabla_x P + \nabla_y Q$ , for any vector  $\vec{V} = (P, Q)$ . In practice, the heat conductivity  $c(x, y, t)$  is defined as function of location gradients. For example, Perona and Malik chose

$$\frac{d\mathbf{I}}{dt} = \nabla_x \left( \frac{1}{1 + (\nabla_x \mathbf{I}/b)^2} \nabla_x \mathbf{I} \right) + \nabla_y \left( \frac{1}{1 + (\nabla_y \mathbf{I}/b)^2} \nabla_y \mathbf{I} \right) \quad (3.34)$$

where  $b$  is a scaling constant. It is easy to see that Equation 3.34 minimizes the following energy function by gradient descent, just as in heat diffusion process in the section above,

$$U(\mathbf{I}) = \int \int \lambda(\nabla_x \mathbf{I}) + \lambda(\nabla_y \mathbf{I}) dx dy \quad (3.35)$$

where  $\lambda(\xi) = \text{const.} \log(1 + (\xi/b)^2)$  and  $\lambda'(\xi) = \text{const.} \frac{\xi}{1 + (\xi/b)^2}$  are plotted in Fig. 3.27.

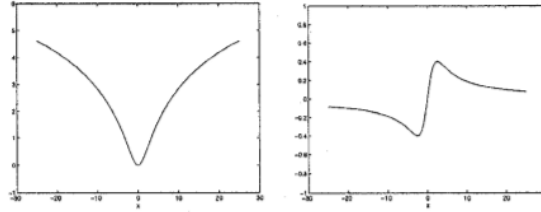


Figure 3.27: Reprinted with permission from [259]. On the left is  $\lambda(\xi) = \text{const.} \log(1 + (\xi/b)^2)$  and on the right is  $\lambda'(\xi) = \text{const.} \frac{\xi}{1 + (\xi/b)^2}$

Although the anisotropic diffusion equations can be adopted for removing noise and enhancing edges [170],  $I(z, y, t)$  converges to a flat image as  $t \rightarrow \infty$  in the Perona-Malik equation.

### 3.6.4 GRADE: Gibbs Reaction And Diffusion Equations

The above sections introduced (very similar) prior PDE paradigms in synthesizing textures. However, the methods mentioned above directly follow PDE models used in chemistry and physics. Directly following Gibbs distribution as used in FRAME, we derive a family of PDEs called Gibbs Reaction and Diffusion Equation (GRADE). We also believe that many PDE paradigms for image processing can be unified under this common framework by using the same approach.

Suppose image  $\mathbf{I}$  is defined on an  $N \times N$  lattice  $D$ . Consider the Gibbs distribution as derived in FRAME,

$$p(\mathbf{I}; \lambda, S) = \frac{1}{Z} \exp \{-U(\mathbf{I}; \Lambda, S)\},$$

where  $Z$  is the normalization constant independent of  $\mathbf{I}$ ,  $S = \{F_1, F_2, \dots, F_n\}$  is a set of filters to characterize the essential features of the images, and  $\Lambda = \{\lambda_1(\cdot), \dots, \lambda_n(\cdot)\}$  is a set of potential functions on feature

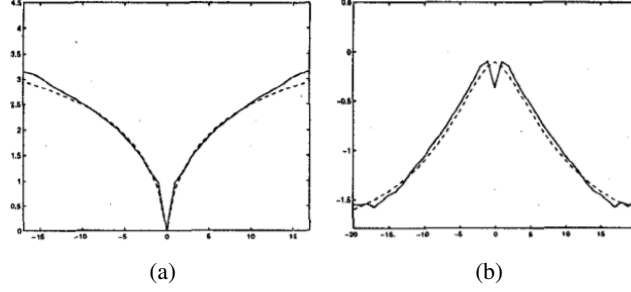


Figure 3.28: Reprinted with permission from [259]. This figure shows two classes of functions. (a) shows diffusion functions. (b) shows reaction functions. Dotted lines show the fitted  $\phi$  and  $\psi$  functions, respectively.

statistics extracted by  $S$  (using convolution). The potential is

$$U(\mathbf{I}; \Lambda, S) = \sum_{i=1}^n \sum_{(x,y) \in D} \lambda_i (F_i * \mathbf{I}(x, y)) \quad (3.36)$$

where  $F_i * \mathbf{I}(x, y)$  is the filter response of  $F_i$  at  $(x, y)$ .

In practice,  $S$  is chosen by minimizing the entropy of  $p(\mathbf{I})$  from a bank of filters such as the Gabor filters at various scales and orientations [3][21] and wavelet transforms [12][20]. In general, these filters can be non-linear functions of the image intensities. In the rest of this section, we shall only study the following linear filters:

1. An intensity filter  $\delta(\cdot)$ , and gradient filters  $\nabla_x, \nabla_y$ .
2. The Laplacian of Gaussian filters,

$$LoG(x, y, s) = const. (x^2 + y^2 - s^2) e^{-\frac{x^2 + y^2}{s^2}}, \quad (3.37)$$

where  $s = \sqrt{2}\sigma$  stands for the scale of the filter. We denote these filters by  $LoG(s)$ . A special filter is  $LoG(\frac{\sqrt{2}}{2})$ , which has a  $3 \times 3$  window  $[0, \frac{1}{4}, 0; \frac{1}{4}, -1, \frac{1}{4}; 0, \frac{1}{4}, 0]$ .

3. Gabor filters with both sine and cosine components:

$$G(x, y, s, \theta) = const. \cdot Rot(\theta) \cdot e^{\frac{1}{2s^2}(4x^2 + y^2)} e^{-i\frac{2\pi}{s}x}. \quad (3.38)$$

It is a sine wave at frequency  $\frac{2\pi}{s}$  modulated by an elongated Gaussian Function, and rotated at angle  $\theta$ . We denote the real and imaginary parts of  $G(z, y, s, \theta)$  by  $G \cos(s, \theta)$  and  $G \sin(s, \theta)$ .

Using the filters above to learn the potential function in Equation 3.36 in discretized bins using FRAME, we found that texture patterns generally exhibit two families of functions similar to reaction-diffusion in chemical processes as shown in Fig. 3.28. We use the following two families of functions to fit our discretized findings, and the fitted curves are shown as dashed lines in Fig. 3.28,

$$\phi(\xi) = a(1 - \frac{1}{1 + (|\xi - \xi_0|/b)^\gamma}), \quad a > 0 \quad (3.39)$$

$$\psi(\xi) = a(1 - \frac{1}{1 + (|\xi - \xi_0|/b)^\gamma}), \quad a < 0 \quad (3.40)$$



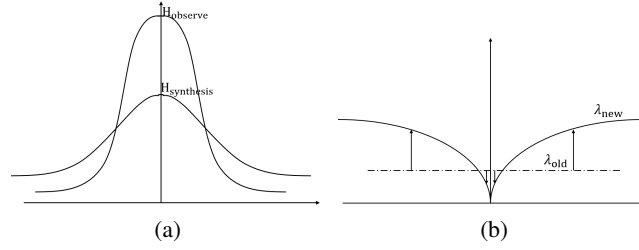


Figure 3.29: Note the forming of the two types of potential functions. Suppose our current histogram of synthesized image is more dispersed than the observed histogram (shown in the left image). Then the updating of  $\lambda$  will push the  $\lambda$  on the tail to be bigger and push the  $\lambda$  in the center to be smaller, which forms a diffusion function. In the following sampling process, this diffusion function will push the corresponding feature to zero, making the synthesized histogram more concentrated to center and thus be closer to the observed one.

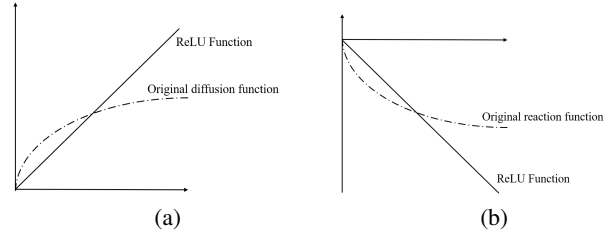


Figure 3.30: We can use the ReLU function to replace the original non-linear diffusion and reaction functions.

where  $\xi_0$ ,  $b$  are the translation and scaling constants, respectively, and  $\|a\|$  weights the contribution of the filter.  $\phi(\xi)$ , the diffusion function, assigns lowest energy (or highest probability) to filter responses around  $\xi_0$  (and  $\xi_0 = 0$  in most cases), and  $\psi(\xi)$ , the reaction function, has lowest energy at the two tails which represent salient image features such as object boundaries. These inverted potential functions are in contrast to all previous image models, and they are essential for modeling realistic images. The forming of the two potential functions is closely related to our training process. Recall Equation 3.21, we have shown take the gradient of  $\lambda$  equal to the difference between the histogram of current synthesized image and the one of our observed image. Then if the synthesized image contains more larger components of a certain feature than the original one. The update will shape the new  $\lambda$  to be more like the diffusion type so that the larger responses are inhibited. On the other hand, if the histogram of synthesized image is too concentrated on certain response, the  $\lambda$  will be shaped to reaction type so that this feature is encouraged to appear. This process is illustrated in Fig. 3.30. Recently, especially in the deep neural network cases, instead of learning a non-linear potential function, people use ReLU function to get similar results. The ReLU function has a linear response in the positive half axis and be zero at negative half axis. Shown in Fig. 3.29, by setting the coefficient of ReLU function to be greater or less than zero, we can get same results of encouraging or preventing a certain feature to appear. We will further discuss this in the later part of Deep FRAME model.

Now we can design our potential function in Equation 3.36 to be

$$U(\mathbf{I}; \Lambda, S) = \sum_{i=1}^{n_d} \sum_{(x,y) \in D} \phi_i(F_i * \mathbf{I}(x, y)) + \sum_{i=n_d}^n \sum_{(x,y) \in D} \phi_i(F_i * \mathbf{I}(x, y)). \quad (3.41)$$

Note that the filter set is divided into two parts  $S = S_d \cup S_r$ , with  $S_d = \{F_i, i = 1, 2, \dots, n_d\}$  and  $S_r = \{F_i, i = n_d + 1, \dots, n\}$ . In most cases,  $S_d$  consists of filters such as gradient filters and Laplacian/Gaussian filters which capture general smoothness appearances of real world images, and  $S_r$  contains filters such as Gabor filters at various orientations and scales which characterize salient features of images.

Our PDE can therefore be designed by maximizing Gibbs distribution, which is equivalent to minimizing the above  $U(\mathbf{I}; \Lambda, S)$  by gradient descent. We obtain the following non-linear parabolic partial differential equation:

$$\frac{d\mathbf{I}}{dt} = \sum_{i=1}^{n_d} F_i^- * \phi'_i(F_i * \mathbf{I}) + \sum_{i=n_d+1}^n F_i^- * \psi'_i(F_i * \mathbf{I}) \quad (3.42)$$

where  $F_i^-(x, y) = -F_i(-x, -y)$ . Thus starting from an input image  $\mathbf{I}(x, y, t) = \mathbf{I}$ , the first term diffuses the image by reducing the gradients while the second term forms patterns as the reaction forces favor large filter responses. We call Equation 3.42 the Gibbs Diffusion And Reaction Equation (GRADE).

### 3.6.5 Properties of GRADE

#### Property 1: A General Statistical Framework

GRADE as in Equation 3.42 can be considered as an extension to the heat diffusion equation, as in Equation 3.33, on a discrete lattice by defining a vector field

$$\vec{V} = (\phi'_1(\cdot), \dots, \phi'_{n_d}(\cdot), \psi'_{n_d+1}(\cdot), \dots, \psi'_n(\cdot))$$

and the divergence operator can be generalized to

$$\text{div} = F_1^- * + \dots + F_n^- * .$$

Thus Equation 3.42 can be written as

$$\frac{d\mathbf{I}}{dt} = \text{div}(\vec{V}). \quad (3.43)$$

Compared to Equation 3.33 which transfers "heat" among adjacent pixels, Equation 3.43 transfers the "heat" in many directions on a graph, and the conductivities are defined as functions of the local patterns instead of just the local gradients. Note that in the discrete lattice, choosing  $S_d = \nabla_x, \nabla_y, S_r = \emptyset$ , we have  $F_1 = F_1^- = \nabla_x, F_2 = F_2^- = \nabla_y$  and  $\text{div} = \nabla_x + \nabla_y$ , thus Equation 3.33 and Equation 3.34 are just special cases of Equation 3.43.

#### Property 2: Diffusion

Fig. 3.31a and Fig. 3.31b show two best-fit potential functions. Fig. 3.31a shows a round tip at  $\xi = 0$ , and Fig. 3.31b shows a cusp at  $\xi = 0$ . Interestingly, real world objects typically show potential function with a cusp. This is because large part of real world objects have flat intensity, encouraging piece-wisely constant regions. Intuitively, with  $\gamma < 1$  and  $\xi = 0$  at location  $(x, y)$ ,  $\phi'(0)$  forms an attractive basin in the neighborhood  $\mathcal{N}_i(x, y)$  specified by filter window  $F_i$  at  $(x, y)$ . For a pixel  $(u, v) \in \mathcal{N}_i(x, y)$ , the depth of attractive basin is  $\|\omega F_i^-(u - x, v - y)\|$ . If a pixel is involved in multiple zero filter responses, it will accumulate the depth of the attractive basin generated by each filter. Thus unless the absolute value of the driving force from other well-defined terms is larger than the total depth of the attractive basin at  $(u, v)$ ,  $\mathbf{I}(u, v)$  will stay unchanged.

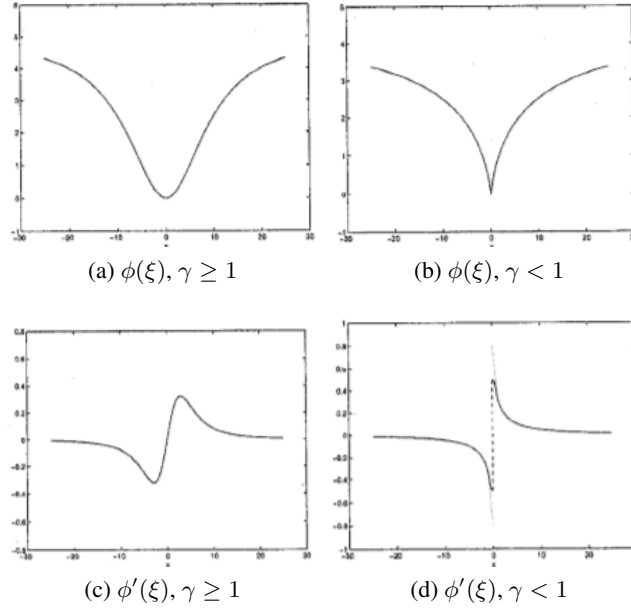


Figure 3.31: Reprinted with permission from [259]. Here  $\phi(\xi) = a(1 - \frac{1}{1+(|\xi-\xi_0|/b)^\gamma})$ ,  $a > 0$ , and its derivative  $\phi'(\xi)$  for a), c) is  $\gamma = 2.0$  and b), d)  $\gamma = 0.8$ .

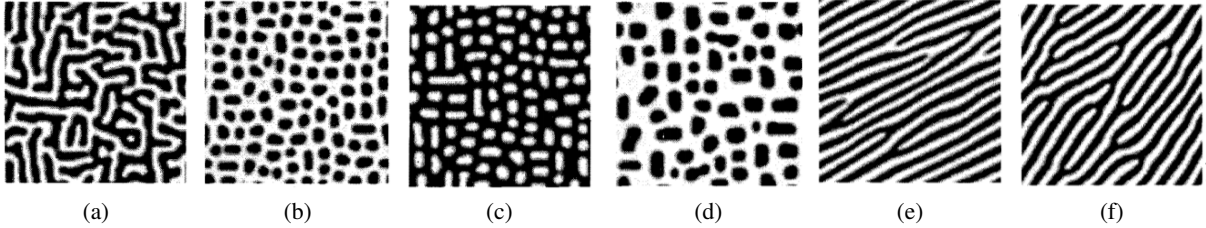


Figure 3.32: Reprinted with permission from [259]. Generated texture patterns.

### Property 3: Reaction

The other class of potential function is reaction function. We refer back to Fig. 3.28 where gradients of reaction function "push" activations away from origin. Different from diffusion function, this class of potential function creates features and brings pixels out of local attraction basins set up by diffusion functions.

With both reaction and diffusion, we can sample a wide variety of textures. Starting from random uniform noise, we use Langevin equations inspired by Brownian motion as below

$$d\mathbf{I} = -\nabla U(\mathbf{I})dt + \sqrt{2T(t)}dw_t \quad (3.44)$$

where  $dw_t = \sqrt{dt}N(0, 1)$ ,  $T$  is the "temperature" which controls magnitude of random fluctuations. In the following figure, we use one diffusion filter, the Laplacian of Gaussian filter, and several reaction filters: isotropic center-surround filters and Gabor filters with different orientations.

## 3.7 Conclusion

In this chapter, we started by asking the question,

*What features and statistics are characteristic of a texture pattern, so that texture pairs that share the same features and statistics cannot be told apart by pre-attentive human visual perception?*

In finding suitable features for texture modeling, we have walked through several models in history including clique-based models and filter-based models. Clique-based models specify their energy function as feature statistics but such models are shown to be inadequate in modeling texture patterns. Filter-based models more closely align with human vision as accepted in neurophysiology. Convolution of filters with an image (filter responses) and histograms of filter activations (marginal statistics projecting an image onto a filter) seem to be a reasonable choice for features.

With features selected, we then introduced Filters, Random field, and Maximum Entropy (FRAME) model as a unifying view over both clique-based and filter-based models. We presented a pursuit process involving Minimax Entropy Principle for pursuing a set of suitable filters and learning suitable potential functions under Gibbs distribution. Texture ensemble is then introduced and connected to statistical physics (micro-canonical ensembles and canonical ensembles). Equivalence between FRAME model and Julesz ensemble on infinite lattice is proved.

Finally, we studied PDE paradigms in synthesizing images and presented two classes of potential functions empirically found in texture modeling - diffusion and reaction. Diffusion is found to destroy features while reaction is found to create features. We also derived our PDE for texture synthesis, introducing Gibbs Reaction and Diffusion Equation (GRADE) as a general statistical framework under which lie many other PDEs used in texture formation.

Here are our attempts on modeling texture patterns and providing a unifying view on texture synthesis. We hope that our works have sparked excitement and passion for further exploration on this matter.

# 4

## Textons

Textons refer to fundamental micro-structures in natural images and are considered as the atoms of pre-attentive human visual perception (Julesz, 1981 [114]). Unfortunately, the term “texton” remains a vague concept in the literature for lacking of a good mathematical model. In this chapter, we present various generative image models for textons.

### 4.1 Distinguishing Textures and Textons

#### 4.1.1 Julesz’s Confusion

In *psychophysics*, Julesz [114] and colleagues discovered that pre-attentive vision is sensitive to some basic image features while ignoring other features. He conjectured that pre-attentive human vision is sensitive to local patterns called textons. Figure 4.1 illustrates the first batch of experiments for texture discrimination.

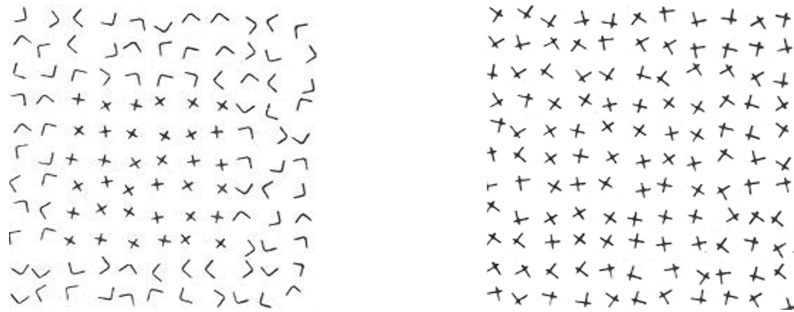


Figure 4.1: Pre-attentive vision is sensitive to local patterns called textons.

In a second batch of experiments, Julesz measured the response time of human subjects in detecting a target element among a number of distractors in the background. For example, Fig. 4.2 shows two pairs of elements in comparison. The response time for the upper pair is instantaneous (100 – 200 *ms*) and independent of the number of distractors. In contrast, for the lower pair the response time increases linearly with the number of distractors. This discovery was very important in psychophysics and motivated Julesz to conjecture a pre-attentive stage that detects some atomic structures, such as elongated blobs, bars, crosses, and terminators, which he called “textons” for the first time.

The early texton studies were limited by their exclusive focus on artificial texture patterns instead of

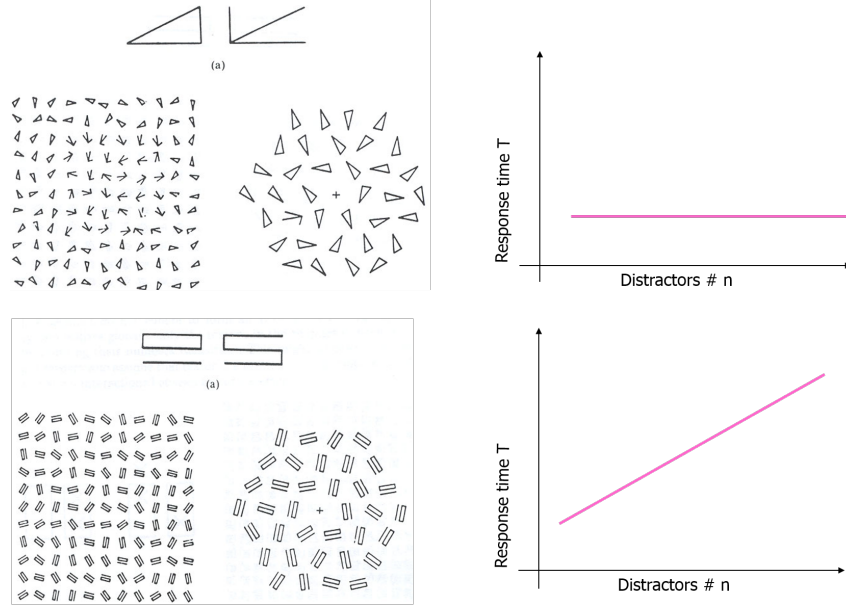


Figure 4.2: Julesz textures show that pre-attentive vision are sensitive to local image structures such as edges, bars, and end points.

natural images. It was shown that the perceptual textons could be adapted through training. Thus the dictionary of textons must be associated with or learned from the ensemble of natural images. Despite the significance of Julesz’s experiments, there have been no rigorous mathematical definitions for textons. Later in this chapter, we argue that textons must be defined in the context of a generative model of images.

In natural images, textures and textons are often interwoven where the image patches are considered to be textures or textons which are respectively from manifolds of different dimensions in the image space. As we have discussed in the previous chapter, the texture patches are from high-dimensional manifolds defined by implicit functions, i.e. the statistical constraints. In this chapter, we will show the texton patches are from low-dimensional manifolds defined by explicit functions, i.e. generative models.

#### 4.1.2 Neural Coding Schemes

Julesz’s experiments are inherently related to how neurons respond to different kinds of stimuli. Studies in neuroscience reveal that neurons propagate signals by generating characteristic electrical pulses called action potentials. With the presence of external stimuli like light and specific image patterns, sensory neurons fire sequences of action potentials in various temporal patterns, in which information about the stimulus is encoded and transmitted to the brain. This process leads to the different perceptions of textures and textons, and justifies the pre-attentive stage proposed by Julesz. Yet, how exactly the neurons represent the signals is still a topic of debate. Here, we present three hypothesized coding schemes and their relation to texton and texture modeling.

Population coding is a method that represents stimuli with the joint activities of multiple neurons. In this coding scheme, each neuron has a distribution of responses over some set of inputs. The responses of many neurons are combined to determine a final value about the inputs, which triggers further reactions in the signal propagation process. In our discussion of image modeling, we can draw parallel between the filters and the neurons. Indeed, the FRAME model combines the potentials of filter responses, i.e., the histogram statistics, and outputs a value to represent an image’s likelihood of belonging to a concept. In this sense,

both the histograms and the neuron responses encode the impression that we have on certain visual stimuli.

Another coding scheme is the grandmother cell coding scheme. In this formulation, high-level concepts are represented by a single neuron. It activates when a person perceives a specific entity, such as one's grandma. To be more concise, there is a single cell in your brain that responds multimodally to your grandmother, granting you the percept of a single entity when you hear the sounds "grandmother", "grandma", and "grams", capture the sight of her smiling, wrinkly face, and recall the tasty cookies she once made as well as the stories she once told you. The hypothesis is supported by an observation in 2004, when an epilepsy patient at the UCLA Medical Center whose brain was being monitored showed vigorous neural responses to several pictures of Jennifer Aniston, but not to other celebrities. Though the theory itself is still controversial, we can nonetheless incorporate it in our study of computer vision. Borrowing the idea of the grandmother cells, we can develop object templates that are invariant under various transformations. These templates, such as cars, represent high-level concepts as a whole rather than individual parts that make up the concept, such as the wheels and front doors of the cars.

The last coding scheme that we are interested in is the sparse coding, where an object is encoded by the strong activation of a relatively small set of neurons. In fact, this method is the foundation of the texton models that we will introduce later in this chapter.

## 4.2 Sparse Coding

### 4.2.1 Image Representation

The purpose of vision, biologic and machine, is to compute a hierarchy of increasingly abstract interpretations of the observed images (or image sequences). Therefore, it is of fundamental importance to know what are the descriptions used at each level of interpretation. By analogy to physics concepts, we wonder what are the visual "electrons", visual "atoms", and visual "molecules" for visual perception. The pursuit of basic images and perceptual elements is not just for intellectual curiosity but has important implications in a series of practical problems. For example,

1. *Dimension reduction.* Decomposing an image into its constituent components reduces information redundancy and leads to lower dimensional representations. As we will show in later examples, an image of  $256 \times 256$  pixels can be represented by about 500 image bases, which are, in turn, reduced to 50-80 texton elements. The dimension of representation is thus reduced by about 100 folds. Further reductions are achieved in motion sequences and lighting models.
2. *Variable decoupling.* The decomposed image elements become more and more independent of each other and thus are spatially nearly decoupled. This facilitates image modeling which is necessary for visual tasks such as segmentation and recognition.
3. *Biologic modeling.* Micro-structures in natural images provide ecological clues for understanding the functions of neurons in early stages of biologic vision systems [9].

In the literature, there are several threads of research investigating fundamental image structures from different perspectives, with many questions left unanswered.

Firstly, in *neurophysiology*, the cells in the early visual pathway (retina, LGN, and V1) of primates are found to compute some basic image structures at various scales and orientations [102]. This motivated some well-celebrated image pyramid representations including Laplacian of Gaussians (LoG), Gabor functions, and their variants. However, very little is known about how V1 cells are grouped into larger structures in

higher levels (say, V2 and V4). Similarly, it is unclear what are the generic image representations beyond the image pyramids in image analysis.

Secondly, in *harmonic analysis*, one treats images as 2D functions, then it can be shown that some classes of functionals (such as Sobolev, Hölder, Besov spaces) can be decomposed into bases, for example, Fourier, wavelets, wedgelets, and ridgelets. It was proven that the Fourier, wavelets, and ridgelets bases are independent components for various functional spaces. But the natural image ensembles known to be very different from those classic mathematical functional spaces.

The third perspective, and the most direct attack to the problem, is the study of *natural image statistics* and *image component analysis*. One important work is done by Olshausen and Field [174] who learned some over-complete image bases from natural image patches ( $12 \times 12$  pixels) with the idea of sparse coding. In contrast to the orthogonal and complete bases in Fourier analysis or tight frame in wavelet transforms, the learned bases are highly correlated, and a given image is coded by a sparse population in the over-complete dictionary. Added to the sparse coding idea is independent component analysis (ICA) which decomposes images into a linear superposition of some image bases minimizing a measure of dependence between the coefficients of these bases [13].

#### 4.2.2 Basis and Frame

In linear algebra, a set  $\mathbf{B}$  of vectors in a vector space  $V$  is called a basis if all elements in  $V$  can be written as a unique linear combination of vectors in  $\mathbf{B}$ . A vector space may have several different bases. Yet all the bases have the same amount of elements equal to the dimension of the vector space. For a typical image with  $1024 \times 1024$  pixels, the dimension of the image space which it belongs to is  $2^{20}$ . Consequently, a basis must have  $2^{20}$  vectors to reconstruct any image of the same size. However, as we have learned earlier, natural images only occupy a small subset of the entire image space, making it trickier to select a set of vectors to represent natural images more efficiently.

If the inner product operation is well defined in a vector space  $V$ , then  $V$  is a inner product space. We call a basis  $\mathbf{B} = \{b_1, b_2, \dots, b_n\}$  for  $V$  orthonormal if the elements are all unit vectors and orthogonal to each other. The Parseval's Identity for orthonormal basis states that

$$\forall x \in V, \|x\|^2 = \sum_{i=1}^n \|\langle x, b_i \rangle\|^2.$$

We can slightly loosen the constraint so that a collection of vectors  $\mathbf{F} = \{f_1, \dots, f_m\}$  satisfies

$$\forall x \in V, A \|x\|^2 \leq \sum_{i=1}^n \|\langle x, f_i \rangle\|^2 \leq B \|x\|^2,$$

then, we call  $\mathbf{F}$  a frame for the vector space with the corresponding frame bounds  $A$  and  $B$ . Furthermore, a frame is a tight frame if  $A = B$ . In finite-dimensional vector spaces, the frames are exactly the spanning sets. Therefore, a vector can be expressed as a linear combination of the frame vectors in a redundant way. Using a frame, it is possible to create a simpler, more sparse representation of a signal compared to representing it strictly with linearly independent vectors.

In general, if a set of vectors is still a basis after removing some elements, then it is called overcomplete. In other words, the number of vectors for a overcomplete basis is greater than the dimension of the input vector. Practically speaking, overcompleteness can help us to achieve a more stable, robust, and compact decomposition of image vectors. We shall see its importance in the sparse coding model of texton representation.



### 4.2.3 Olshausen-Field Model

In image coding, one starts with a dictionary of base functions

$$\Psi = \{\psi_\ell(u, v), \ell = 1, \dots, L_\psi\}.$$

For example, some commonly used bases are Gabor, Laplacian-of-Gaussian (LoG), and other wavelet transforms.

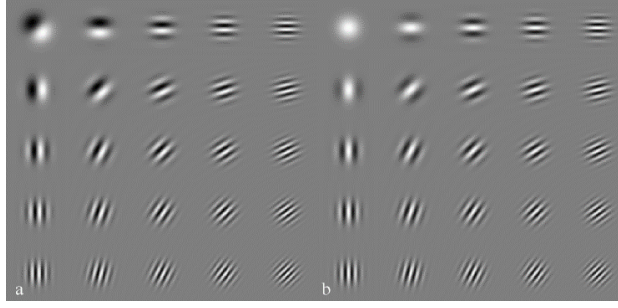


Figure 4.3: Gabor wavelets are sine and cosine waves multiplied by Gaussian functions.

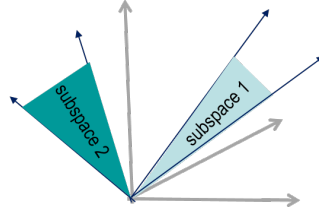


Figure 4.4: The sparse coding model assumes that the observed signals lie on the low dimensional subspaces spanned by the basis vectors.

Let  $A = (x, y, \tau, \sigma)$  denote the translation, rotation and scaling transform of a base function, and  $G_A \ni A$  the orthogonal transform space (group), then we obtain a set of image bases  $\Delta$ ,

$$\Delta = \{\psi_\ell(u, v, A) : A = (x, y, \tau, \sigma) \in G_A, \ell = 1, \dots, L_\psi\}.$$

A simple *generative image model*, adopted in almost all image coding schemes, assumes that an image  $\mathbf{I}$  is a linear superposition of some image bases selected from  $\Delta$  plus a Gaussian noise image  $\mathbf{n}$ .

$$\mathbf{I} = \sum_i^{n_B} \alpha_i \cdot \psi_i + \mathbf{n}, \quad \psi_i \in \Delta, \forall i, \quad (4.1)$$

where  $n_B$  is the number of bases and  $\alpha_i$  is the *coefficient* of the  $i$ -th base  $\psi_i$ .

As  $\Delta$  is over-complete\*, the variables  $(\ell_i, \alpha_i, x_i, y_i, \tau_i, \sigma_i)$  indexing a base  $\psi_i$  are treated as latent (hidden) variables and must be inferred probabilistically, in contrast to deterministic transforms such as the Fourier transform. All the hidden variables are summarized in a *base map*,

$$\mathbf{B} = (n_B, \{b_i = (\ell_i, \alpha_i, x_i, y_i, \tau_i, \sigma_i) : i = 1, 2, \dots, n_B\}).$$

---

\*The number of bases in  $\Delta$  is often 100 times larger than the number of pixels in an image.

If we view each base  $\psi_i$  as an attributed point with attributes  $b_i = (\ell_i, \alpha_i, x_i, y_i, \tau_i, \sigma_i)$ , then  $\mathbf{B}$  is an attributed spatial point process.

In the image coding literature, the bases are assumed to be independently and identically distributed (iid), and the locations, scales and orientations are assumed to be uniformly distributed, so

$$p(\mathbf{B}) = p(n_B) \prod_{i=1}^{n_B} p(b_i), \quad (4.2)$$

$$p(b_i) = p(\alpha_i) \cdot \text{unif}(\ell_i) \cdot \text{unif}(x_i, y_i) \cdot \text{unif}(\tau_i) \cdot \text{unif}(\sigma_i). \quad (4.3)$$

It was well-known that responses of image filters on natural images have high kurtosis histograms. This means that most of the time the filters have nearly zero response (i.e. they are silent) and they are activated with large response occasionally. This leads to the sparse coding idea by Olshausen and Field [174].<sup>†</sup> For example,  $p(\alpha)$  is chosen to be a Laplacian distribution, or a mixture of two Gaussians with  $\sigma_1$  close to zero. For all

$$i = 1, \dots, n_B,$$

$$p(\alpha_i) \sim \exp\{-|\alpha_i|/c\} \quad \text{or} \quad p(\alpha_i) = \sum_{j=1}^2 \omega_j N(0, \sigma_j).$$

In fact, as long as  $p(\alpha)$  has high kurtosis, the exact form of  $p(\alpha)$  is not so crucial. For example, one can choose a mixture of two uniform distributions on a range  $[-\sigma_j, \sigma_j]$ ,  $j = 1, 2$  respectively, with  $\sigma_1$  close to zero,

$$p(\alpha_i) = \sum_{j=1}^2 \omega_j \text{unif}[-\sigma_j, \sigma_j].$$

In the literature a common misconception is that the sparse coding scheme assumes  $n_B = |\Delta|$ , i.e. all bases in the set are “activated”. The prior  $p(\alpha)$  is supposed to suppress most of the activations to zero. It is simple to prove that this is equivalent to assume  $p(\alpha)$  to be a uniform distribution and put a penalty to the model complexity, for example  $p(n_B) \propto e^{-\lambda n_B}$ . So the sparse coding prior is essentially a model complexity term.

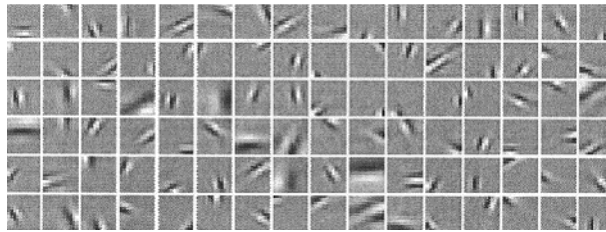


Figure 4.5: Reprinted with permission from [174]. The above image patches are the basis vectors learned from natural image patches by the sparse coding model.

In the above image model, the base map  $\mathbf{B}$  includes the hidden variables and the dictionary  $\Psi$  are parameters. For example, Olshausen and Field used  $L_\psi = 144$  base functions, each being a  $12 \times 12$  matrix. Following an EM-learning algorithm, they learned  $\Psi$  from a large number of natural image patches. Fig. 4.5

<sup>†</sup>Note that the filter responses are convolutions of a filter with image in a deterministic way, and are different from the coefficients of the bases.

shows some of the 144 base functions. Such bases capture some image structures and are believed to bear resemblance to the responses of simple cells in V1 of primates.

In their experiments, the training images are chopped into  $12 \times 12$  pixel patches, therefore they didn't really infer the hidden variables for the transformation  $A_i$ . Thus the learned bases are not aligned at centers and are rather noisy.

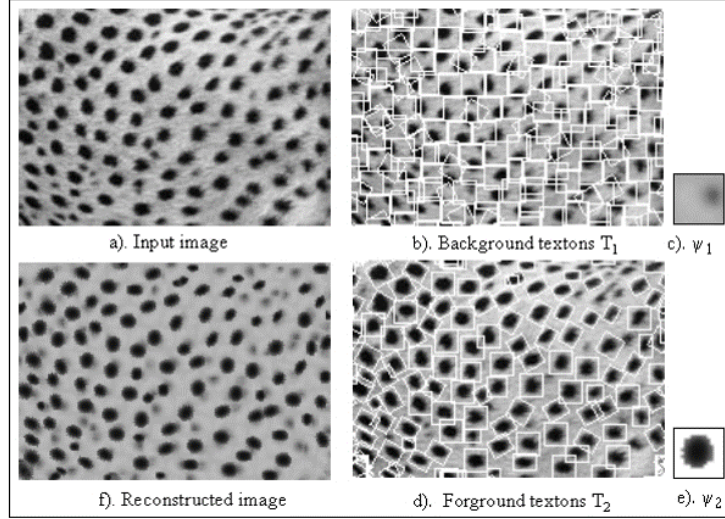


Figure 4.6: Reprinted with permission from [80]. The observed image can be reconstructed from foreground and background textons.

#### 4.2.4 A Three-Level Generative Model

We may extend the previously introduced simple form of a generative image model to a three-level generative model as shown in Fig. 4.12. In this model, an image  $\mathbf{I}$  is generated by a base map  $\mathbf{B}$  as in image coding, and the bases are selected from a dictionary  $\Psi$  with some orthogonal transforms.

The base map  $\mathbf{B}$  is, in turn, generated by a texton map  $\mathbf{T}$ . The texton elements are selected from a texton dictionary  $\Pi$  with some orthogonal transforms. Each texton element in  $\mathbf{T}$  consists of a few bases with a deformable geometric configuration. So we have,

$$\mathbf{T} \xrightarrow{\Pi} \mathbf{B} \xrightarrow{\Psi} \mathbf{I},$$

with

$$\Psi = \{\psi_\ell, \ell = 1, 2, \dots, L_\psi\}, \quad \text{and} \quad \Pi = \{\pi_\ell; \ell = 1, 2, \dots, L_\pi\}.$$

By analogy to the *waveform-phoneme-word* hierarchy in speech, the *pixel-base-texton* hierarchy presents an increasingly abstract visual description. This representation leads to dimension reduction and the texton elements account for spatial co-occurrence of the image bases.

To clarify terminology, a base function  $\psi \in \Psi$  is like a mother wavelet and an image base  $b_i$  in the base map  $\mathbf{B}$  is an instance under certain transforms of a base function. Similarly, a “texton” in a texton dictionary  $\pi \in \Pi$  is a deformable template, while a “texton element” is an instance in the texton map  $\mathbf{T}$  which is a transformed and deformed version of a texton in  $\Pi$ .

For natural images, it is reasonable to guess that the number of base functions is about  $|\Psi| = O(10)$ , and the number of textons is in the order of  $|\Pi| = O(10^3)$  for various combinations. Intuitively, textons

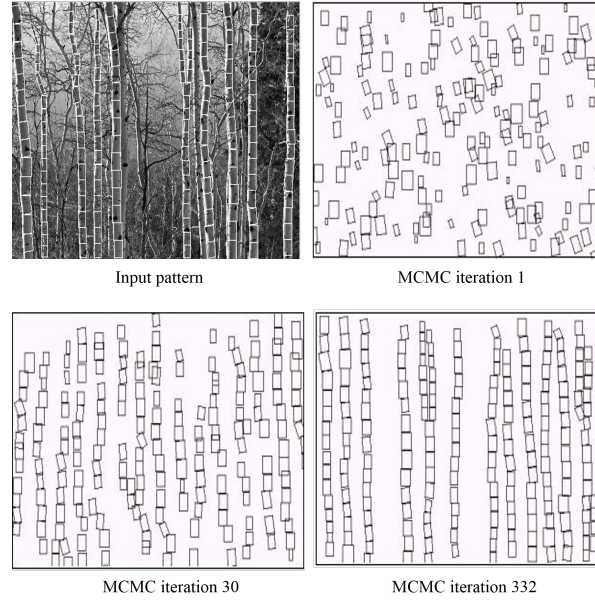


Figure 4.7: Reprinted with permission from [80]. The spatial arrangement of textons can be modeled by a point process model.

are meaningful objects viewed at distance (i.e. small scale), such as stars, birds, cheetah blobs, snowflakes, beans, etc.

In this chapter, we fix the base dictionary to three common base functions: Laplacian-of-Gaussian, Gabor cosine, Gabor sine, i.e.,

$$\Psi = \{ \psi_1, \psi_2, \psi_3 \} = \{ \text{LoG}, \text{Gcos}, \text{Gsin} \}.$$

These base functions are not enough for patterns like hair or water, etc. But we fix them for simplicity and focus on the learning of texton dictionary  $\mathbf{\Pi}$ . This chapter is also limited to learning textons for each individual texture pattern instead of generic natural images, therefore  $|\mathbf{\Pi}|$  is a small number for each texture.

Before we formulate the problem, we show an example of simple star pattern to illustrate the generative texton model. In Fig. 4.13, we first show the three base functions in  $\Psi$  (the first row) and their symbolic sketches. Then for an input image, a matching pursuit algorithm is adopted to compute the base map  $\mathbf{B}$  in a bottom-up fashion. This base map will be modified later by stochastic inference. It is generally observed that the base map  $\mathbf{B}$  can be divided into two sub-layers. One sub-layer has relatively large (“heavy”) coefficients  $\alpha_i$  and captures some larger image structures. For the star pattern these are the LoG bases shown in the first column. We show both the symbolic sketch of these LoG bases (above) and the image generated by these bases (below). The heavy bases are usually surrounded by a number of “light” bases with relatively small coefficients  $\alpha_i$ . We put these secondary bases in another sub-layer (see the second column of Fig. 4.13). When these image bases are superpositioned, they generate a reconstructed image (see the third column in Fig. 4.13). The residues of reconstruction are assumed to be Gaussian noise.

By an analogy to physics model, we call the heavy bases the “nucleus bases” as they have heavy weights like protons and neutrons, and the light bases the “electron bases”. Fig. 4.9 displays an “atomic” model for the star texton. It is a LoG base surrounded by 5 electron bases.

The three-level generative model is governed by a joint probability specified with parameters  $\Theta =$

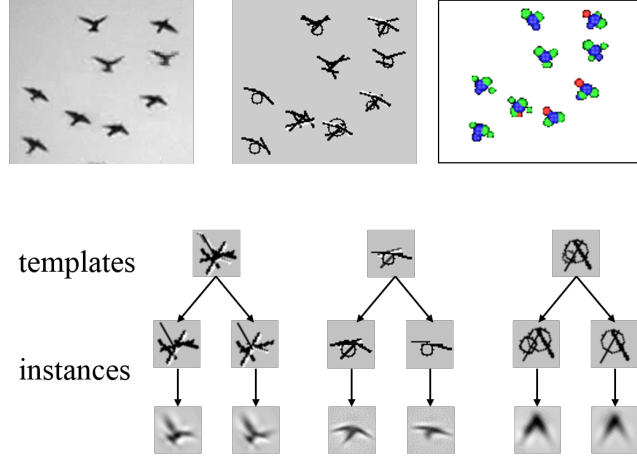


Figure 4.8: Reprinted with permission from [225]. A texton template is a deformable composition of basis vectors.

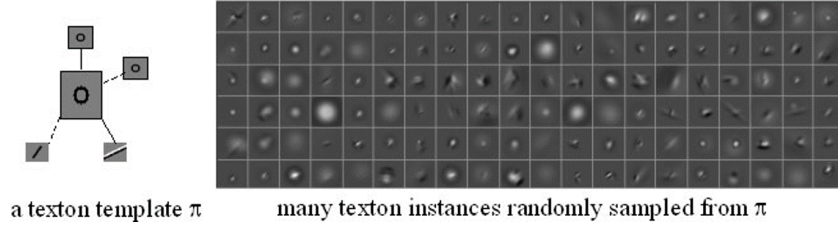


Figure 4.9: Reprinted with permission from [225]. A texton template is a deformable composition of basis vectors.

$(\Psi, \Pi, \kappa)$ .

$$p(\mathbf{I}, \mathbf{B}, \mathbf{T}; \Theta) = p(\mathbf{I}|\mathbf{B}; \Psi)p(\mathbf{B}|\mathbf{T}; \Pi)p(\mathbf{T}; \kappa),$$

where  $\Psi$  and  $\Pi$  are dictionaries for two generating processes, and  $p(\mathbf{T}; \kappa)$  is a descriptive (Gibbs) model for the spatial distribution of the textons as a stochastic attributed point process.

We rewrite the base map as

$$\mathbf{B} = (n_B, \{b_i = (\ell_i, \alpha_i, x_i, y_i, \tau_i, \sigma_i) : i = 1, 2, \dots, n_B\}). \quad (4.4)$$

Because we assume Gaussian distribution  $N(0, \sigma_o^2)$  for the reconstruction residues, we have

$$p(\mathbf{I}|\mathbf{B}; \Psi) \propto \exp\left\{-\sum_{(u,v) \in D} (\mathbf{I}(u,v) - \sum_{i=1}^{n_B} \alpha_i \psi_{\ell_i}(u,v; x_i, y_i, \tau_i, \sigma_i))^2 / 2\sigma_o^2\right\}. \quad (4.5)$$

The  $n_B$  bases in base map  $\mathbf{B}$  are divided into  $n_T + 1$  groups ( $n_T < n_B$ ).

$$\{b_i = (\ell_i, \alpha_i, x_i, y_i, \tau_i, \sigma_i) : i = 1, 2, \dots, n_B\} = \varpi_0 \cup \varpi_1 \cup \dots \cup \varpi_{n_T}.$$

Bases in  $\varpi_0$  are “free electrons” which do not belong to any texton, and are subject to the independent distribution  $p(b_j)$  in equation (4.3). Bases in any other class form a texton element  $T_j$ , and the texton map is

$$\mathbf{T} = (n_T, \{T_j = (\ell_j, \alpha_j, x_j, y_j, \tau_j, \sigma_j, \delta_j) : j = 1, 2, \dots, n_T\}).$$

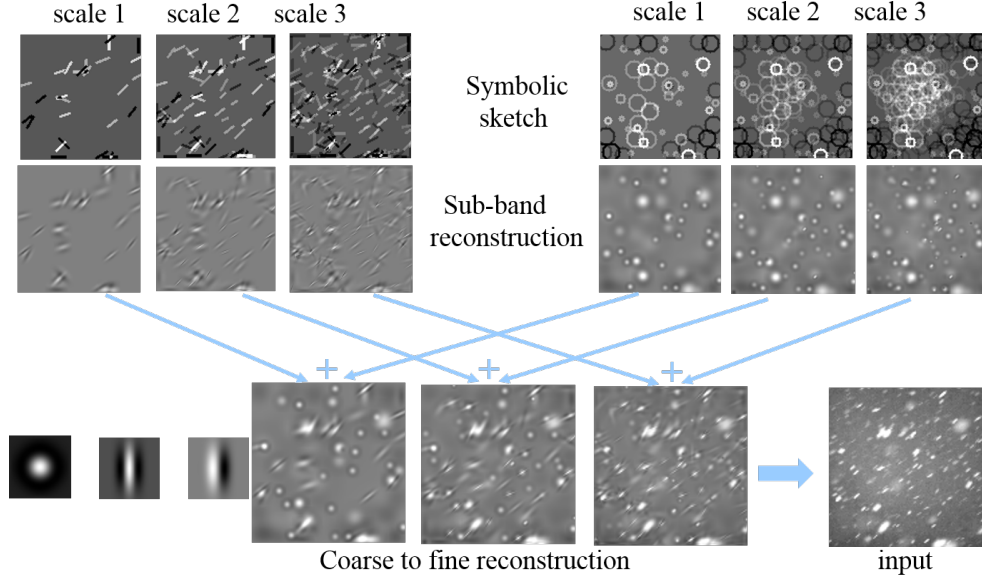


Figure 4.10: Reprinted with permission from [225]. The image can be decomposed into sub-bands at multiple scales.

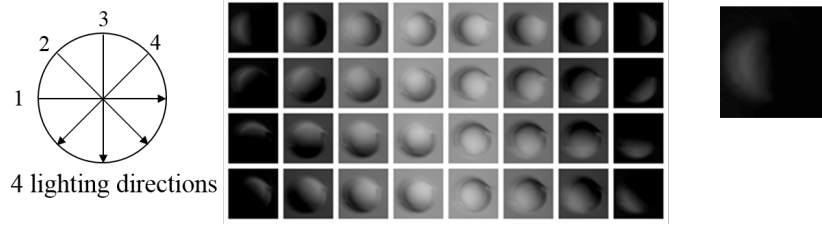


Figure 4.11: Reprinted with permission from [258]. The concept of textons can also be generalize to incorporate lighting variations.

Each texton element  $T_j$  is specified by its type  $\ell_j$ , photometric contrast  $\alpha_j$ , translation  $(x_j, y_j)$ , rotation  $\tau_j$ , scaling  $\sigma_j$  and deformation vector  $\delta_j$ . A texton  $\pi \in \Pi$  consists of  $m$  image bases with a certain deformable configuration

$$\pi = ( (\ell_1, \alpha_1, \tau_1, \sigma_1), (\ell_2, \alpha_2, \delta x_2, \delta y_2, \delta \tau_2, \delta \sigma_2), \dots, (\ell_m, \alpha_m, \delta x_m, \delta y_m, \delta \tau_m, \delta \sigma_m) ).$$

The  $(\delta x, \delta y, \delta \tau, \delta \sigma)$  are the relative positions, orientations and scales. Therefore, we have

$$p(\mathbf{B}|\mathbf{T}; \Pi) = p(|\varpi_0|) \prod_{b_j \in \varpi_0} p(b_j) \prod_{c=1}^{n_T} p(\varpi_c | T_c; \pi_{\ell_c}).$$

$p(\mathbf{T}; \kappa)$  is another distribution which accounts for the number of textons  $n_T$  and the spatial relationship among them. It can be a Gibbs model for attributed point process. For simplicity, we assume the textons are independent at this moment as a special Gibbs model.

By integrating out the hidden variables, we obtain a likelihood probability for any observable image  $\mathbf{I}^{\text{obs}}$ ,

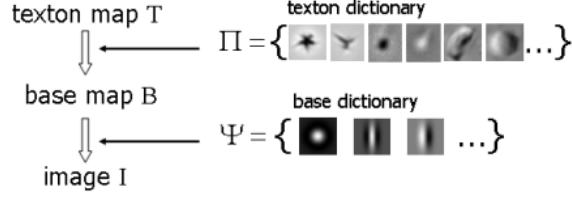


Figure 4.12: Reprinted with permission from [258]. A three-level generative model: an image  $\mathbf{I}$  is a linear addition of some image bases selected from a base dictionary  $\Psi$ , such as Gabor or Laplacian of Gaussians. The base map is further generated by a smaller number of textons selected from a texton dictionary  $\Pi$ . Each texton consists of a number of bases in certain deformable configurations.

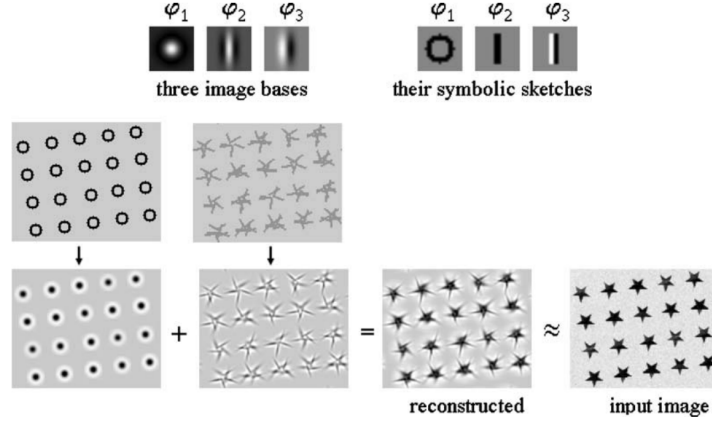


Figure 4.13: Reprinted with permission from [258]. Reconstructing a star pattern by two layers of bases. An individual star is decomposed into a LoG base in the upper layer for the body of the star plus a few other bases (mostly Gcos, Gsin) in the lower layer for the angles.

$$p(\mathbf{I}^{\text{obs}}; \Theta) = \int p(\mathbf{I}^{\text{obs}} | \mathbf{B}; \Psi) p(\mathbf{B} | \mathbf{T}; \Pi) p(\mathbf{T}; \kappa) d\mathbf{B} d\mathbf{T}.$$

In  $p(\mathbf{I}; \Theta)$  above, the parameters  $\Theta$  (dictionaries, etc.) characterize the entire image ensemble, like the vocabulary for English or Chinese languages. In contrast, the hidden variables  $\mathbf{B}$ ,  $\mathbf{T}$  are associated with an individual image  $\mathbf{I}$ , and correspond to the parsing tree in language.

Our goal is to learn the parameters  $\Theta = (\Psi, \Pi, \kappa)$  by maximum likelihood estimation, or equivalently minimizing a Kullback-Leibler divergence between a underlying probability of images  $f(\mathbf{I})$  and  $p(\mathbf{I}; \Theta)$

$$\Theta^* = (\Psi, \Pi, \kappa)^* = \arg \min \text{KL}(f(\mathbf{I}) || p(\mathbf{I}; \Theta)) = \arg \max \sum_m \log p(\mathbf{I}_m^{\text{obs}}; \Theta) + \epsilon \quad (4.6)$$

where  $\epsilon$  is an approximation error which diminishes as sufficient data are available for training. In practice,  $\epsilon$  may decide the complexity of the models, and thus the number of base functions  $L_\psi$  and textons  $L_\pi$ . For clarity, we use only one large  $\mathbf{I}^{\text{obs}}$  for training, because multiple images can be considered just patches of a larger image. For motion and lighting models in later sections,  $\mathbf{I}^{\text{obs}}$  is extended to image sequence and image set with illumination variations.



By fitting the generative model to observed images, we can learn the texton dictionary as parameters of the generative model.

With such model, we may study the *geometric*, *dynamic*, and *photometric* structures of the texton representation by further extending the generative model to account for motion and illumination variations. 1) For the geometric structures, a texton consists of a number of image bases with deformable spatial configurations. The geometric structures are learned from static texture images. 2) For the dynamic structures, the motion of a texton is characterized by a Markov chain model in time which sometimes can switch geometric configurations during the movement. We call the moving textons as “motons”. The dynamic models are learned using the trajectories of the textons inferred from a video sequence. 3) For photometric structures, a texton represents the set of images of a 3D surface element under varying illuminations and is called a “lighton”. We adopt an illumination-cone representation where a lighton is a texton triplet. For a given light source, a lighton image is generated as a linear sum of the three texton bases.

## 4.3 Active Basis Model

### 4.3.1 Olshausen-Field Model for Sparse Coding

The active basis model is based on the sparse coding model of Olshausen and Field (1996). Olshausen and Field proposed that the role of simple V1 cells is to compute sparse representations of natural images. Let  $\{\mathbf{I}_m, m = 1, \dots, M\}$  be a set of small image patches. For example, they might be  $12 \times 12$  patches, in which case  $\mathbf{I}_m \in \mathbb{R}^{12 \times 12}$ . We may think of each  $\mathbf{I}_m$  as a two-dimensional function defined on the  $12 \times 12$  lattice. The Olshausen-Field model seeks to represent these images by

$$\mathbf{I}_m = \sum_{i=1}^N c_{m,i} B_i + U_m, \quad (4.7)$$

where  $(B_i, i = 1, \dots, N)$  is a dictionary of basis functions defined on the same image lattice (e.g.,  $12 \times 12$ ) as  $\mathbf{I}_m$ ,  $c_{m,i}$  are the coefficients, and  $U_m$  is the unexplained residual image.  $N$  is often assumed to be greater than the number of pixels in  $\mathbf{I}_m$ , so the dictionary is said to be over-complete and is therefore redundant. However, the number of coefficients  $(c_{m,i}, i = 1, \dots, N)$  that are non-zero (or significantly different from zero) is assumed to be small (e.g., less than 10) for each image  $\mathbf{I}_m$ .

One may also assume that the basis functions in the dictionary are translated, rotated and dilated versions of one another, so that each  $B_i$  can be written as  $B_{x,s,\alpha}$ , where  $x$  is the location (a two-dimensional vector),  $s$  is the scale, and  $\alpha$  is the orientation. We call such a dictionary self-similar, and we call  $(x, s, \alpha)$  the geometric attribute of  $B_{x,s,\alpha}$ .

Model (4.7) then becomes

$$\mathbf{I}_m = \sum_{x,s,\alpha} c_{m,x,s,\alpha} B_{x,s,\alpha} + U_m, \quad (4.8)$$

where  $B_{x,s,\alpha}$  are translated, rotated and dilated copies of a single basis function, e.g.,  $B = B_{x=0,s=1,\alpha=0}$ , and  $(x, s, \alpha)$  are properly discretized (default setting:  $\alpha$  is discretized into 16 equally spaced orientations).  $B$  can be learned from training images  $\{\mathbf{I}_m\}$ .

From now on, we assume that the dictionary of basis functions is self-similar, and  $\{B_{x,s,\alpha}, \forall (x, s, \alpha)\}$  is already given. In the following, we assume that  $B_{x,s,\alpha}$  is a Gabor wavelet, and we also assume that  $B_{x,s,\alpha}$



is normalized to have unit  $\ell_2$  norm so that  $|B_{x,y,\alpha}|^2 = 1$ .  $B_{x,s,\alpha}$  may also be a pair of Gabor sine and cosine wavelets, so that for each Gabor wavelet  $B$ ,  $B = (B_0, B_1)$ . The corresponding coefficient  $c = (c_0, c_1)$ , and  $cB = c_0B_0 + c_1B_1$ . The projection  $\langle \mathbf{I}, B \rangle = (\langle \mathbf{I}, B_0 \rangle, \langle \mathbf{I}, B_1 \rangle)$ , and  $|\langle \mathbf{I}, B \rangle|^2 = \langle \mathbf{I}, B_0 \rangle^2 + \langle \mathbf{I}, B_1 \rangle^2$ .

Given the dictionary  $(B_{x,s,\alpha}, \forall(x, s, \alpha))$ , the encoding of an image  $\mathbf{I}_m$  amounts to inferring  $(c_{m,x,s,\alpha}, \forall(x, s, \alpha))$  in (4.8) under the sparsity constraint, which means that only a small number of  $(c_{m,x,s,\alpha})$  are non-zero. That is, we seek to encode  $\mathbf{I}_m$  by

$$\mathbf{I}_m = \sum_{i=1}^n c_{m,i} B_{x_{m,i}, s_{m,i}, \alpha_{m,i}} + U_m, \quad (4.9)$$

where  $n \ll N$  is a small number, and  $(x_{m,i}, s_{m,i}, \alpha_{m,i}, i = 1, \dots, n)$  are the geometric attributes of the selected basis functions whose coefficients  $(c_{m,i})$  are non-zero. The attributes  $(x_{m,i}, s_{m,i}, \alpha_{m,i}, i = 1, \dots, n)$  form a spatial point process (we continue to use  $i$  to index the basis functions, but here  $i$  only runs through the  $n$  selected basis functions instead of all the  $N$  basis functions as in (4.7)).

### 4.3.2 Active Basis Model for Shared Sparse Coding of Aligned Image Patches

The active basis model was proposed for modeling deformable templates formed by basis functions.

Suppose we have a set of training image patches  $\{\mathbf{I}_m, m = 1, \dots, M\}$ . This time we assume that they are defined on the same bounding box, and the objects in these images come from the same category. In addition, these objects appear at the same location, scale and orientation, and in the same pose. See Figure 4.14 for 9 image patches of deer. We call such image patches aligned.

The active basis model is of the following form

$$\mathbf{I}_m = \sum_{i=1}^n c_{m,i} B_{x_i + \Delta x_{m,i}, s, \alpha_i + \Delta \alpha_{m,i}} + U_m, \quad (4.10)$$

where  $\mathbf{B} = (B_{x_i, s, \alpha_i}, i = 1, \dots, n)$  form the nominal template of an active basis model (sometimes we simply call  $\mathbf{B}$  an active basis template). Here we assume that the scale  $s$  is fixed and given.  $\mathbf{B}_m = (B_{x_i + \Delta x_{m,i}, s, \alpha_i + \Delta \alpha_{m,i}}, i = 1, \dots, n)$  is the deformed version of the nominal template  $\mathbf{B}$  for encoding  $\mathbf{I}_m$ , where  $(\Delta x_{m,i}, \Delta \alpha_{m,i})$  are the perturbations of the location and orientation of the  $i$ -th basis function from its nominal location  $x_i$  and nominal orientation  $\alpha_i$  respectively. The perturbations are introduced to account for shape deformation. Both  $\Delta x_{m,i}$  and  $\Delta \alpha_{m,i}$  are assumed to vary within limited ranges (default setting:  $\Delta x_{m,i} \in [-3, 3]$  pixels, and  $\Delta \alpha_{m,i} \in \{-1, 0, 1\} \times \pi/16$ ).

### 4.3.3 Prototype Algorithm

Given the dictionary of basis functions  $\{B_{x,s,\alpha}, \forall(x, s, \alpha)\}$ , the learning of the active basis model from the aligned image patches  $\{\mathbf{I}_m\}$  involves the sequential selection of  $B_{x_i, s, \alpha_i}$  and the inference of its perturbed version  $B_{x_i + \Delta x_{m,i}, s, \alpha_i + \Delta \alpha_{m,i}}$  in each image  $\mathbf{I}_m$ . We call the learning as supervised, because the bounding boxes of the objects are given and the images are aligned. See Figure 4.14 for an illustration of the learning results.

In this subsection, we consider a prototype version of the shared matching pursuit algorithm, which is to be revised in the following subsections. The reason we start from this prototype algorithm is that it is simple and yet captures the key features of the learning algorithm.

We seek the maximal reduction of the least squares reconstruction error in each iteration (recall that the basis functions are normalized to have unit  $\ell_2$  norm):

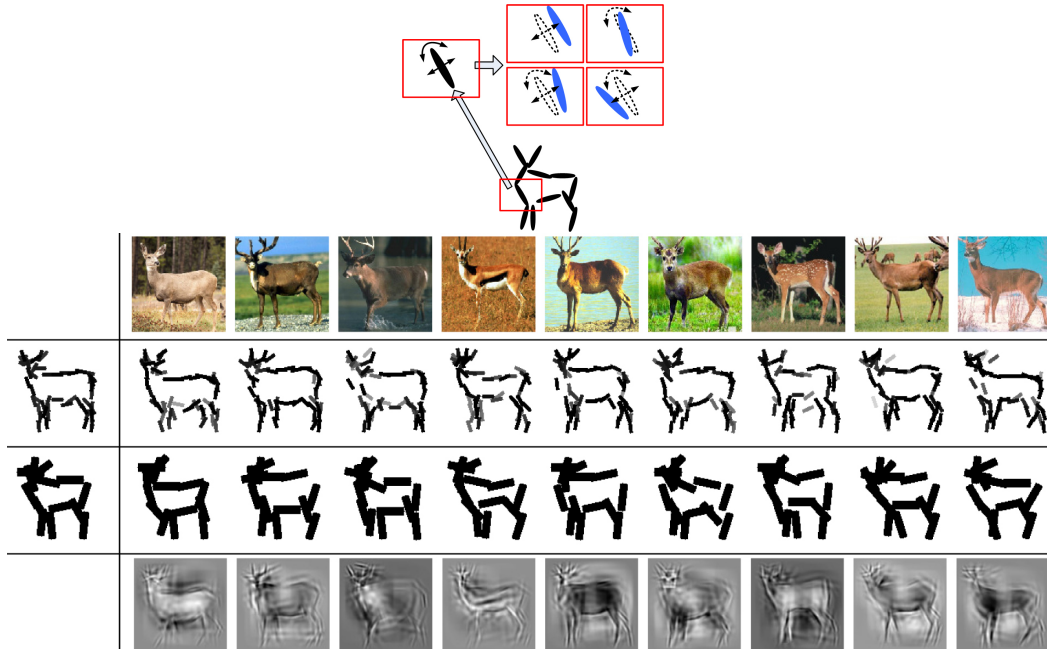


Figure 4.14: Reprinted with permission from [96]. (a) An active basis model is a composition of a small number of basis functions, such as Gabor wavelets at selected locations and orientations. Each basis function can perturb its location and orientation within limited ranges. (b) Supervised learning of active basis model from aligned images. In this example, two active basis models are learned using Gabor wavelets at two different scales (there is no variation in the aspect ratio of the Gabor wavelets used). The first row displays the 9 training images. The second row: the first plot is the nominal template formed by 50 basis functions. The rest of the plots are the deformed templates matched to the images by perturbing the basis functions. The third row: the same as the second row, except that the scale of the Gabor wavelets is about twice as large, and the number of wavelets is 14. The last row displays the linear reconstruction of each training image from 100 selected and perturbed basis functions.

$$\sum_{m=1}^M |\mathbf{I}_m - \sum_{i=1}^n c_{m,i} B_{x_i + \Delta x_{m,i}, s, \alpha_i + \Delta \alpha_{m,i}}|^2. \quad (4.11)$$

The prototype algorithm is a greedy algorithm which minimizes the reconstruction error:

0. Initialize  $i \leftarrow 0$ . For  $m = 1, \dots, M$ , initialize the residual image  $U_m \leftarrow \mathbf{I}_m$ .
1.  $i \leftarrow i + 1$ . Select the next basis function by

$$(x_i, \alpha_i) = \arg \max_{x, \alpha} \sum_{m=1}^M \max_{\Delta x, \Delta \alpha} |\langle U_m, B_{x + \Delta x, s, \alpha + \Delta \alpha} \rangle|^2,$$

where  $\max_{\Delta x, \Delta \alpha}$  is local maximum pooling within the small ranges of  $\Delta x_{m,i}$  and  $\Delta \alpha_{m,i}$ .

2. For  $m = 1, \dots, M$ , given  $(x_i, \alpha_i)$ , infer the perturbations in location and orientation by retrieving the arg-max in the local maximum pooling of step 1:

$$(\Delta x_{m,i}, \Delta \alpha_{m,i}) = \arg \max_{\Delta x, \Delta \alpha} |\langle U_m, B_{x_i + \Delta x, s, \alpha_i + \Delta \alpha} \rangle|^2. \quad (4.12)$$

Let  $c_{m,i} \leftarrow \langle U_m, B_{x_i + \Delta x_{m,i}, s, \alpha_i + \Delta \alpha_{m,i}} \rangle$ , and update the residual image by explaining away:

$$U_m \leftarrow U_m - c_{m,i} B_{x_i + \Delta x_{m,i}, s, \alpha_i + \Delta \alpha_{m,i}}. \quad (4.13)$$

3. Stop if  $i = n$ , else go back to step 1.

#### Algorithm 4: Prototype Algorithm

Simultaneous (or collaborative) sparse approximation of multiple signals has been proposed in harmonic analysis and signal processing literature.

In Equation (4.13), the perturbed basis function  $B_{x_i + \Delta x_{m,i}, s, \alpha_i + \Delta \alpha_{m,i}}$  explains away part of  $U_m$ . As a result, nearby basis functions that overlap with  $B_{x_i + \Delta x_{m,i}, s, \alpha_i + \Delta \alpha_{m,i}}$  tend not to be selected in future iterations. So the basis functions selected for each deformed template  $\mathbf{B}_m = (B_{x_i + \Delta x_{m,i}, s, \alpha_i + \Delta \alpha_{m,i}}, i = 1, \dots, n)$  usually have little overlap with each other. For computational and modeling convenience, we shall assume that these selected basis functions are orthogonal to each other, so that the coefficients can be obtained by projection:  $c_{m,i} = \langle \mathbf{I}_m, B_{x_i + \Delta x_{m,i}, s, \alpha_i + \Delta \alpha_{m,i}} \rangle$ .

Correspondingly, the explaining-away step can then be carried out by local inhibition. Specifically, after we identify the perturbed basis function  $B_{x_i + \Delta x_{m,i}, s, \alpha_i + \Delta \alpha_{m,i}}$ , we simply prohibit nearby basis functions that are correlated with  $B_{x_i + \Delta x_{m,i}, s, \alpha_i + \Delta \alpha_{m,i}}$  from being included in the deformed template  $\mathbf{B}_m$ . In practice, we allow small correlations between the basis functions in each  $\mathbf{B}_m$ .

#### 4.3.4 Statistical Modeling

The above algorithm guided by (4.11) implicitly assumes that the unexplained background image  $U_m$  is Gaussian white noise. This assumption can be problematic because the unexplained background may contain salient structures such as edges, and the Gaussian white noise distribution clearly cannot account for such structures. This is why we need to revise the above algorithm which is based on the Gaussian white

noise assumption. A better assumption is to assume that  $U_m$  follows the same distribution as that of natural images.

More precisely, the distribution of  $\mathbf{I}_m$  given the deformed template  $\mathbf{B}_m = (B_{x_i+\Delta x_{m,i},s,\alpha_i+\Delta\alpha_{m,i}}, i = 1, \dots, n)$ , i.e.,  $p(\mathbf{I}_m | \mathbf{B}_m)$ , is obtained by modifying the distribution of natural images  $q(\mathbf{I}_m)$  in such a way that we only change the distribution of  $C_m = (c_{m,i} = \langle \mathbf{I}_m, B_{x_i+\Delta x_{m,i},s,\alpha_i+\Delta\alpha_{m,i}} \rangle, i = 1, \dots, n)$  from  $q(C_m)$  to  $p(C_m)$ , while leaving the conditional distribution of  $U_m$  given  $C_m$  unchanged. Here  $p(C_m)$  and  $q(C_m)$  are the distributions of  $C_m$  under  $p(\mathbf{I}_m | \mathbf{B}_m)$  and  $q(\mathbf{I}_m)$  respectively. Thus the model is in the form of foreground  $p(C_m)$  popping out from background  $q(\mathbf{I}_m)$ . Specifically,  $p(\mathbf{I}_m | \mathbf{B}_m) = q(\mathbf{I}_m)p(C_m)/q(C_m)$ .

The reason for such a form is as follows.  $C_m$  is the projection of  $\mathbf{I}_m$  into  $\mathbf{B}_m$ . Let  $U_m$  be the projection of  $\mathbf{I}_m$  into the remaining subspace that is orthogonal to  $\mathbf{B}_m$ . Then  $p(\mathbf{I}_m | \mathbf{B}_m)/q(\mathbf{I}_m) = p(C_m, U_m)/q(C_m, U_m) = p(C_m)/q(C_m)$ . The second equality follows from the assumption that  $p(U_m|C_m) = q(U_m|C_m)$ , i.e., we keep the conditional distribution of  $U_m$  given  $C_m$  fixed.

For computational simplicity, we further assume  $(c_{m,i} = \langle \mathbf{I}_m, B_{x_i+\Delta x_{m,i},s,\alpha_i+\Delta\alpha_{m,i}} \rangle, i = 1, \dots, n)$  are independent given  $\mathbf{B}_m$ , under both  $p$  and  $q$ , so

$$p(\mathbf{I}_m | \mathbf{B}_m) = q(\mathbf{I}_m) \prod_{i=1}^n \frac{p_i(c_{m,i})}{q(c_{m,i})},$$

where  $q(c)$  is assumed to be the same for  $i = 1, \dots, n$  because  $q(\mathbf{I}_m)$  is translation and rotation invariant.  $q(c)$  can be pooled from natural images in the form of a histogram of Gabor filter responses. This histogram is heavy-tailed because of the edges in natural images.

For parametric modeling, we model  $p_i(c_{m,i})/q(c_{m,i})$  in the form of exponential family model. Specifically, we assume the following exponential family model  $p_i(c) = p(c; \lambda_i)$ , which is in the form of exponential tilting of the reference distribution  $q(c)$ :

$$p(c; \lambda) = \frac{1}{Z(\lambda)} \exp\{\lambda h(|c|^2)\} q(c), \quad (4.14)$$

so that  $p(c; \lambda)/q(c)$  is in the exponential form. We assume  $\lambda_i > 0$ , and  $h(r)$  is a sigmoid-like function of the response  $r = |c|^2$  that saturates for large  $r$  (recall that the Gabor filter response  $c = (c_0, c_1)$  consists of responses from the pair of Gabor sine and cosine wavelets, and  $|c|^2 = c_0^2 + c_1^2$ ). Specifically, we assume that  $h(r) = \xi[2/(1 + e^{-2r/\xi}) - 1]$ , so  $h(r) \approx r$  for small  $r$ , and  $h(r) \rightarrow \xi$  as  $r \rightarrow \infty$  (default setting:  $\xi = 6$ ). The reason we want  $h(r)$  to approach a fixed constant for large  $r$  is that there can be strong edges in both the foreground and background, albeit with different frequencies.  $p(c; \lambda)/q(c)$  should approach the ratio between these two frequencies for large  $r = |c|^2$ . In (4.14),

$$Z(\lambda) = \int \exp\{\lambda h(r)\} q(c) dc = E_q[\exp\{\lambda h(r)\}]$$

is the normalizing constant.

$$\mu(\lambda) = E_\lambda[h(r)] = \int h(r) p(c; \lambda) dc$$

is the mean parameter. Both  $Z(\lambda)$  and  $\mu(\lambda)$  can be computed beforehand from a set of natural images.

The exponential family model can be justified by the maximum entropy principle. Given the deformed template  $\mathbf{B}_m = (B_{x_i+\Delta x_{m,i},s,\alpha_i+\Delta\alpha_{m,i}}, i = 1, \dots, n)$ , consider the coefficients obtained by projection:  $(c_{m,i}(\mathbf{I}_m) = \langle \mathbf{I}_m, B_{x_i+\Delta x_{m,i},s,\alpha_i+\Delta\alpha_{m,i}} \rangle, i = 1, \dots, n)$ . Suppose we want to find a probability distribution  $p(\mathbf{I}_m | \mathbf{B}_m)$  so that  $E[h(|c_{m,i}(\mathbf{I}_m)|^2)] = \mu_i$  for some fixed  $\mu_i, i = 1, \dots, n$ , where  $\mu_i$  can be estimated from

the training images. Then among all the distributions that satisfy the constraints on  $E[h(|c_{m,i}(\mathbf{I}_m)|^2)]$ , the distribution that is closest to  $q(\mathbf{I}_m)$  in terms of the Kullback-Leibler divergence is given by

$$p(\mathbf{I}_m | \mathbf{B}_m) = \frac{1}{Z(\Lambda)} \exp\left\{\sum_{i=1}^n \lambda_i h(|c_{m,i}(\mathbf{I}_m)|^2)\right\} q(\mathbf{I}_m),$$

where  $\Lambda = (\lambda_i, i = 1, \dots, n)$ ,  $Z(\Lambda) = E_q[\exp\{\sum_{i=1}^n \lambda_i h(|c_{m,i}(\mathbf{I}_m)|^2)\}]$  is the normalizing constant, and  $\Lambda$  is chosen to satisfy the constraints on  $E[h(|c_{m,i}(\mathbf{I}_m)|^2)]$ . If we further assume that  $c_{m,i}(\mathbf{I}_m)$  are independent of each other for  $i = 1, \dots, n$  under  $q(\mathbf{I}_m)$ , then  $c_{m,i}(\mathbf{I}_m)$  are also independent under  $p(\mathbf{I}_m | \mathbf{B}_m)$ , and their distributions are of the form (4.14).

In order to choose the nominal template  $\mathbf{B}$  and the deformed templates  $\{\mathbf{B}_m, m = 1, \dots, M\}$ , we want  $p(\mathbf{I}_m | \mathbf{B}_m)$  to be farthest from  $q(\mathbf{I}_m)$  in terms of Kullback-Leibler divergence. From a classification point of view, we want to choose  $\mathbf{B}$  and  $\{\mathbf{B}_m\}$  so that the features  $\{h(|c_{m,i}|^2), i = 1, \dots, n\}$  lead to the maximal separation between training images (e.g., images of deer) and generic natural images.

The log-likelihood ratio between the current model  $p(\mathbf{I}_m | \mathbf{B}_m)$  and the reference model  $q(\mathbf{I}_m)$  is

$$l(\{\mathbf{I}_m\} | \mathbf{B}, \{\mathbf{B}_m\}, \Lambda) = \sum_{m=1}^M \log \frac{p(\mathbf{I}_m | \mathbf{B}_m)}{q(\mathbf{I}_m)} \quad (4.15)$$

$$= \sum_{m=1}^M \sum_{i=1}^n [\lambda_i h(|\langle \mathbf{I}_m, B_{x_i + \Delta x_{m,i}, s, \alpha_i + \Delta \alpha_{m,i}} \rangle|^2) - \log Z(\lambda_i)] . \quad (4.16)$$

The expectation of the above log-likelihood ratio is the Kullback-Leibler divergence between  $p(\mathbf{I}_m | \mathbf{B}_m)$  and  $q(\mathbf{I}_m)$ .

Given the training images  $\{\mathbf{I}_m, m = 1, \dots, M\}$ ,  $\sum_{m=1}^M \log q(\mathbf{I}_m)$  is a constant. Thus maximizing the log-likelihood ratio  $\sum_{m=1}^M \log p(\mathbf{I}_m | \mathbf{B}_m, \Lambda) / q(\mathbf{I}_m)$  is equivalent to maximizing the log-likelihood  $\sum_{i=1}^M \log p(\mathbf{I}_m | \mathbf{B}_m, \Lambda)$ . So in the following, with a slight abuse of terminology, we occasionally refer to the log-likelihood ratio as the log-likelihood.

### 4.3.5 Shared Matching Pursuit

We revise the prototype algorithm in subsection (4.3.3) so that each iteration seeks the maximal increase of the log-likelihood ratio (4.16) instead of the maximum reduction of the least squares reconstruction error (4.11) as in subsection (4.3.3). The revised version of the shared matching pursuit algorithm is as follows.

0. Initialize  $i \leftarrow 0$ . For  $m = 1, \dots, M$ , initialize the response maps  $R_m(x, \alpha) \leftarrow \langle \mathbf{I}_m, B_{x,s,\alpha} \rangle$  for all  $(x, \alpha)$ .

1.  $i \leftarrow i + 1$ . Select the next basis function by finding

$$(x_i, \alpha_i) = \arg \max_{x, \alpha} \sum_{m=1}^M \max_{\Delta x, \Delta \alpha} h(|R_m(x + \Delta x, \alpha + \Delta \alpha)|^2),$$

where  $\max_{\Delta x, \Delta \alpha}$  is again local maximum pooling.

2. For  $m = 1, \dots, M$ , given  $(x_i, \alpha_i)$ , infer the perturbations by retrieving the arg-max in the local maximum pooling of step 1:

$$(\Delta x_{m,i}, \Delta \alpha_{m,i}) = \arg \max_{\Delta x, \Delta \alpha} |R_m(x_i + \Delta x, \alpha_i + \Delta \alpha)|^2.$$

Let  $c_{m,i} \leftarrow R_m(x_i + \Delta x_{m,i}, \alpha_i + \Delta \alpha_{m,i})$ , and update  $R_m(x, \alpha) \leftarrow 0$  if the correlation

$$\text{corr}[B_{x,s,\alpha}, B_{x_i+\Delta x_{m,i},s,\alpha_i+\Delta \alpha_{m,i}}] > \epsilon$$

(default setting:  $\epsilon = .1$ ). Then compute  $\lambda_i$  by solving the maximum likelihood equation  $\mu(\lambda_i) = \sum_{m=1}^M h(|c_{m,i}|^2)/M$ .

3. Stop if  $i = n$ , else go back to step 1.

### Algorithm 5: Revised Prototype Algorithm

For each candidate  $(x_i, \alpha_i)$ , the maximum likelihood equation  $\mu(\lambda_i) = \sum_{m=1}^M h(|c_{m,i}|^2)/M$  is obtained by taking the derivative of the log-likelihood ratio, where  $\mu(\lambda_i) = \mathbb{E}_{\lambda_i}[h(|c|^2)] = \int h(|c|^2)p(c; \lambda_i)dc$  is the mean parameter, and is a monotonically increasing function of  $\lambda_i > 0$ . So its inverse  $\mu^{-1}()$  is also a monotonically increasing function.  $\lambda_i$  is solved so that  $\mu(\lambda_i)$  matches the empirical average of  $h(|c_{m,i}|^2)$ ,  $m = 1, \dots, M$ . The function  $\mu()$  can be computed and stored over a discrete set of equal-spaced values, so that  $\lambda_i$  can be solved by looking up these values with linear interpolations between them.

Because  $h()$  is monotonically increasing, the maximized log-likelihood ratio is monotone in the estimated  $\lambda_i$ . The estimated  $\lambda_i$  is in turn monotone in the average  $\sum_{m=1}^M h(|c_{m,i}|^2)/M$ . So the maximized log-likelihood ratio is monotone in  $\sum_{m=1}^M h(|c_{m,i}|^2)/M$ . Therefore, in step [1],  $(x_i, \alpha_i)$  is chosen by maximizing the sum  $\sum_{m=1}^M \max_{\Delta x, \Delta \alpha} h(|R_m(x + \Delta x, \alpha + \Delta \alpha)|^2)$  over all possible  $(x, \alpha)$ .

In step [2], the arg-max basis function inhibits nearby basis functions to enforce the approximate orthogonality constraint. The correlation is defined as the square of the inner product between the basis functions and can be computed and stored beforehand.

After learning the template from training images  $\{\mathbf{I}_m\}$ , we can use the learned template to detect the object in a testing image  $\mathbf{I}$ .

1. For every pixel  $X$ , compute the log-likelihood ratio  $l(X)$ , which serves as the template matching score at putative location  $X$ :

$$l(X) = \sum_{i=1}^n [\lambda_i \max_{\Delta x, \Delta \alpha} h(|\langle \mathbf{I}, B_{X+x_i+\Delta x, s, \alpha_i+\Delta \alpha} \rangle|^2) - \log Z(\lambda_i)]. \quad (4.17)$$

2. Find maximum likelihood location  $\hat{X} = \arg \max_X l(X)$ . For  $i = 1, \dots, n$ , inferring perturbations by retrieving the arg-max in the local maximum pooling in step [1]:

$$(\Delta x_i, \Delta \alpha_i) = \arg \max_{\Delta x, \Delta \alpha} |\langle \mathbf{I}, B_{\hat{X}+x_i+\Delta x, s, \alpha_i+\Delta \alpha} \rangle|^2.$$

3. Return the location  $\hat{X}$ , and  $(B_{\hat{X}+x_i+\Delta x_i, s, \alpha_i+\Delta \alpha_i}, i = 1, \dots, n)$ , which is the translated and deformed template.

#### Algorithm 6: Object Detection

*Rotation and multi-resolution.* We can rotate the template and scan the template over multiple resolutions of the original image, to account for uncertainties about the orientation and scale of the object in the testing image.

### 4.3.6 Active Appearance Models

In this sub-section, we briefly review a method of interpreting images using an Active Appearance Model (AAM) proposed in [27]. The first AAM model was actually introduced by Peter Hallinan at Harvard, it was later written in a book [83]. An AAM contains a statistical model of the shape and grey-level appearance of the object of interest which can generalise to almost any valid example. During a training phase we learn the relationship between model parameter displacements and the residual errors induced between a training image and a synthesised model example. To match to an image [27] measure the current residuals and use the model to predict changes to the current parameters, leading to a better fit. A good overall match is obtained in a few iterations, even from poor starting estimates. The AAM algorithm is an important method for locating deformable objects in many applications. In AAM, the image difference patterns corresponding to changes in each model parameter are learnt and used to modify a model estimate.

In a parallel development Sclaroff and Isidoro have demonstrated 'Active Blobs' for tracking [196]. The approach is broadly similar in that they use image differences to drive tracking, learning the relationship between image error and parameter offset in an off-line processing stage. The main difference is that Active Blobs are derived from a single example, whereas Active Appearance Models use a training set of examples. The former use a single example as the original model template, allowing deformations consistent with low energy mesh deformations (derived using a Finite Element method). A simply polynomial model is used to allow changes in intensity across the object. AAMs learn what are valid shape and intensity variations from their training set.

First, appearance models are generated follows the algorithms described in [45] but includes extra normalisation and weighting steps. The models were generated by combining a model of shape variation with a model of the appearance variations in a shape-normalised frame. We require a training set of labelled images, where key landmark points are marked on each example object. For instance, to build a face model we require face images marked with points at key positions to outline the main features.

Given such a set we can generate a statistical model of shape variation). The labelled points on a single object describe the shape of that object. We align all the sets into a common co-ordinate frame and represent each by a vector,  $\mathbf{x}$ . We then apply a principal component analysis (PCA) to the data. Any example can then be approximated using:  $\mathbf{x} = \bar{\mathbf{x}} + \mathbf{P}_s \mathbf{b}_s$ , where  $\bar{\mathbf{x}}$  is the mean shape,  $\mathbf{P}_s$  is a set of orthogonal modes of variation and  $\mathbf{b}_s$  is a set of shape parameters.

To build a statistical model of the grey-level appearance we warp each example image so that its control points match the mean shape (using a triangulation algorithm). We then sample the grey level information  $\mathbf{g}_{im}$  from the shape-normalised image over the region covered by the mean shape. To minimise the effect of global lighting variation, we normalise the example samples by applying a scaling,  $\alpha$ , and offset,  $\beta$ ,  $\mathbf{g} = (\mathbf{g}_m - \beta \mathbf{1})/\alpha$ . The values of  $\alpha$  and  $\beta$  are chosen to best match the vector to the normalised mean. Let  $\bar{\mathbf{g}}$  be the mean of the normalised data, scaled and offset so that the sum of elements is zero and the variance of elements is unity. The values of  $\beta$  and  $\alpha$  required to normalise  $\mathbf{g}_{im}$  are then given by  $\alpha = \mathbf{g}_{im} \cdot \bar{\mathbf{g}}, \beta = (\mathbf{g}_{im} \cdot \mathbf{1})/n$ , where  $n$  is the number of elements in the vectors.

Of course, obtaining the mean of the normalised data is then a recursive process, as the normalisation is defined in terms of the mean. A stable solution can be found by using one of the examples as the first estimate of the mean, aligning the others to it (using 2 and 3), re-estimating the mean and iterating. By applying PCA to the normalised data we obtain a linear model:  $\mathbf{g} = \bar{\mathbf{g}} + \mathbf{P}_g \mathbf{b}_g$ , where  $\bar{\mathbf{g}}$  is the mean normalised grey-level vector,  $\mathbf{P}_g$  is a set of orthogonal modes of variation and  $\mathbf{b}_g$  is a set of grey-level parameters. The shape and appearance of any example can thus be summarised by the vectors  $\mathbf{b}_s$  and  $\mathbf{b}_g$ . Since there may be correlations between the shape and grey-level variations, we apply a further PCA to the data as follows. For each example we generate the concatenated vector  $\mathbf{b} = [\mathbf{W}_s \mathbf{b}_s, \mathbf{b}_g]^T = [\mathbf{W}_s \mathbf{P}_s^T (\mathbf{x} - \bar{\mathbf{x}}), \mathbf{P}_g^T (\mathbf{g} - \bar{\mathbf{g}})]^T$ , where  $\mathbf{W}_s$  is a diagonal matrix of weights for each shape parameter, allowing for the difference in units between the shape and grey models. We apply a PCA on these vectors, giving a further model  $\mathbf{b} = \mathbf{Q} \mathbf{c}$ , where  $\mathbf{Q}$  are the eigenvectors and  $\mathbf{c}$  is a vector of appearance parameters controlling both the shape and grey-levels of the model. Since the shape and grey-model parameters have zero mean,  $\mathbf{c}$  does too. Note that the linear nature of the model allows us to express the shape and grey-levels directly as functions of  $\mathbf{c}$ ,  $\mathbf{x} = \bar{\mathbf{x}} + \mathbf{P}_s \mathbf{W}_s \mathbf{Q}_s \mathbf{c}$ ,  $\mathbf{g} = \bar{\mathbf{g}} + \mathbf{P}_g \mathbf{Q}_g \mathbf{c}$ , where  $\mathbf{Q} = [\mathbf{Q}_s, \mathbf{Q}_g]^T$ .

An example image can be synthesised for a given  $\mathbf{c}$  by generating the shape-free grey-level image from the vector  $\mathbf{g}$  and warping it using the control points described by  $\mathbf{x}$ .

We now address the central problem: We have an image to be interpreted, a full appearance model as described above and a reasonable starting approximation. [27] propose a scheme for adjusting the model parameters efficiently, so that a synthetic example is generated, which matches the new image as closely as possible.

We wish to treat interpretation as an optimisation problem in which we minimise the difference between a new image and one synthesised by the appearance model. A difference vector  $\delta \mathbf{I}$  can be defined as  $\delta \mathbf{I} = \mathbf{I}_i - \mathbf{I}_m$ , where  $\mathbf{I}_i$  is the vector of grey-level values in the image, and  $\mathbf{I}_m$ , is the vector of grey-level values for the current model parameters. To locate the best match between model and image, we wish to minimise the magnitude of the difference vector,  $\Delta = |\delta \mathbf{I}|^2$ , by varying the model parameters,  $\mathbf{c}$ . Since the appearance models can have many parameters, this appears at first to be a difficult high-dimensional optimisation problem. We note, however, that each attempt to match the model to a new image is actually a similar optimisation problem. We propose to learn something about how to solve this class of problems in advance. By providing a-priori knowledge of how to adjust the model parameters during image search, we arrive at an efficient runtime algorithm. In particular, the spatial pattern in  $\delta \mathbf{I}$ , encodes information about how the model parameters should be changed in order to achieve a better fit. In adopting this approach there are two parts to the problem: learning the relationship between  $\delta \mathbf{I}$  and the error in the model parameters,  $\delta \mathbf{c}$  and using this knowledge in an iterative algorithm for minimising  $\Delta$ .



The simplest model we could choose for the relationship between  $\delta\mathbf{I}$  and the error in the model parameters (and thus the correction which needs to be made) is linear:  $\delta\mathbf{c} = \mathbf{A}\delta\mathbf{I}$ . This turns out to be a good enough approximation to achieve acceptable results. To find  $\mathbf{A}$ , we perform multiple multivariate linear regression on a sample of known model displacements,  $\delta\mathbf{c}$ , and the corresponding difference images,  $\delta\mathbf{I}$ . We can generate these sets of random displacements by perturbing the 'true' model parameters for the images in which they are known. These can either be the original training images or synthetic images generated with the appearance model. In the latter case we know the parameters exactly, and the images are not corrupted by noise.

As well as perturbations in the model parameters, we also model small displacements in 2D position, scale, and orientation. These four extra parameters are included in the regression; for simplicity of notation, they can be regarded simply as extra elements of the vector  $\delta\mathbf{c}$ . To retain linearity we represent the pose using  $(s_x, s_y, t_x, t_y)$  where  $s_x = s\cos(\theta)$ ,  $s_y = s\sin(\theta)$ . In order to obtain a well-behaved relationship it is important to choose carefully the frame of reference in which the image difference is calculated.

We calculate a difference thus: Let  $\mathbf{c}_0$  be the known appearance model parameters for the current image. We displace the parameters by a known amount,  $\delta\mathbf{c}$ , to obtain new parameters  $\mathbf{c} = \delta\mathbf{c} + \mathbf{c}_0$ . For these parameters we generate the shape,  $\mathbf{x}$ , and normalised grey-levels,  $\mathbf{g}_\mathbf{in}$ , using  $\mathbf{x} = \bar{\mathbf{x}} + \mathbf{P}_s \mathbf{W}_s \mathbf{Q}_s \mathbf{c}$ ,  $\mathbf{g} = \bar{\mathbf{g}} + \mathbf{P}_g \mathbf{Q}_g \mathbf{c}$ , where  $\mathbf{Q} = [\mathbf{Q}_s, \mathbf{Q}_g]^T$ . We sample from the image, warped using the points,  $\mathbf{x}$ , to obtain a normalised sample  $\mathbf{g}_\mathbf{s}$ . The sample error is then  $\delta\mathbf{g} = \mathbf{g} - \mathbf{g}_\mathbf{m}$ .

The training algorithm is then simply to randomly displace the model parameter in each training image, recording  $\delta\mathbf{c}$  and  $\delta\mathbf{g}$ . We then perform multi-variate regression to obtain the relationship  $\delta\mathbf{c} = \mathbf{A}\delta\mathbf{g}$ .

The best range of values of  $\delta\mathbf{c}$  to use during training is determined experimentally. Ideally we seek to model a relationship that holds over as large a range errors,  $\delta\mathbf{g}$ , as possible. However, the real relationship is found to be linear only over a limited range of values. Our experiments on the face model suggest that the optimum perturbation was around 0.5 standard deviations (over the training set) for each model parameter, about 10% in scale and 2 pixels translation.

Given a method for predicting the correction which needs to be made in the model parameters we can construct an iterative method for solving our optimisation problem. Given the current estimate of model parameters,  $\mathbf{c}_0$ , and the normalised image sample at the current estimate,  $\mathbf{g}_\mathbf{s}$ , one step of the iterative procedure is as follows:

- Evaluate the error vector  $\delta\mathbf{g}_0 = \mathbf{g}_\mathbf{s} - \mathbf{g}_\mathbf{m}$
- Evaluate the error  $E_0 = |\delta\mathbf{g}_0|^2$
- Compute the predicted displacement,  $\delta\mathbf{c} = \mathbf{A}\delta\mathbf{g}$
- Set  $k = 1$
- Let  $\mathbf{c}_1 = \mathbf{c}_0 - k\delta\mathbf{c}$
- Sample the image at this new prediction, and calculate a new error vector  $\delta\mathbf{g}_1$
- If  $|\delta\mathbf{g}_1|^2 < E_0$  then accept the new estimate  $\mathbf{c}_1$
- Otherwise try at  $k = 1.5, k = 0.5, k = 2.5$ , etc.

This procedure is repeated until no improvement is made to the error,  $|\delta\mathbf{g}_0|^2$ , and convergence is declared. [27] use a multi-resolution implementation, in which we iterate to convergence at each level before projecting the current solution to the next level of the model. This is more efficient and can converge to the correct solution from further away than search at a single resolution.

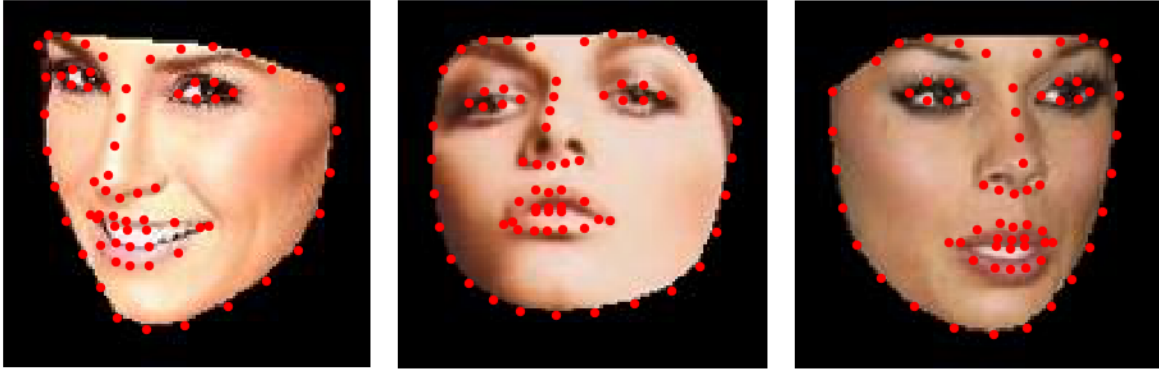


Figure 4.15: Example faces with 68 landmarks from CelebA. The data set contains 1000 images from CelebA, and they are cropped to  $128 \times 128$  pixels by the OpenFace, so that the background and hair are removed. These faces have different colors, illuminations, identities, viewing angles, shapes, and expressions.

The AAM combines modeling of image appearance and its variability with modeling of shape and its variability. To model the image shape and its variability, the AAM needs pre-labeled landmark points, e.g. Figure. 4.15, then the AAM is built by warping the training shapes to the average shape. Face warping with sparse landmark points usually consists of three steps: (i) Find corresponding points. (e.g.) in our dataset, there are 68 landmark points plus 4 corner points. (ii) Define a triangular mesh over the points. The meshes of the source to target image are the same, and we have triangle-to-triangle correspondences. A triangulation of set of points in the plane is a partition of the convex hull to triangles whose vertices are the points, and do not contain other points. There are an exponential number of triangulations of a point set, while the Delaunay triangulation is the "best". Given a set of points in a plane, a Voronoi diagram partitions the space such that the boundary lines are equidistant from neighboring points. If you connect the points in neighboring Voronoi regions, you get a Delaunay triangulation, e.g., Figure 4.16, (iii) Warp each triangle

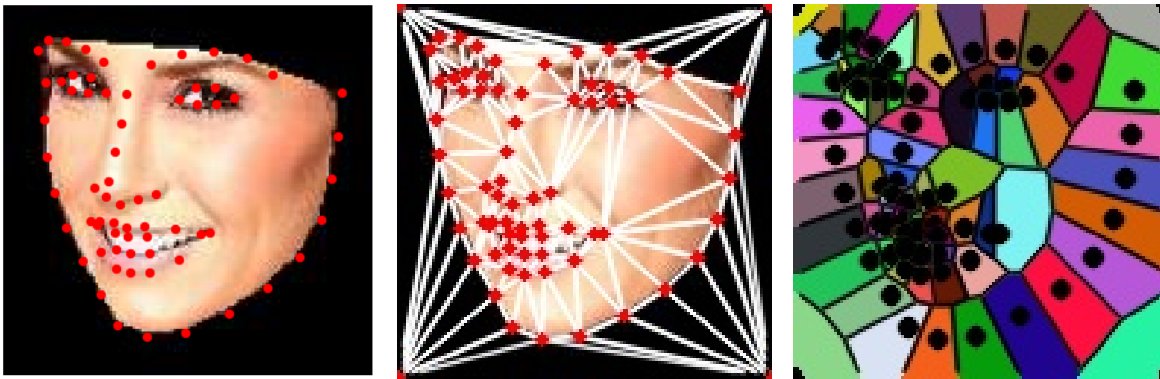


Figure 4.16: Left: Sample face from CelebA with landmarks detected using OpenFace. Center: Delaunay triangulation of the landmarks. Right : Corresponding Voronoi Diagram.

separately from source to target. First, pick a triangle in the source image and the corresponding triangle in the target image and calculate the affine transform that maps the three corners of the triangle in the source image to the three corners of the corresponding triangle in the target image. Then use the calculated affine

transform to transform all pixels inside the triangle to the target image. Repeat this for all triangles in the source image to obtain a warped version of it.

In the context of compressing the face images and communicate through a network with small number of bits, AAM can reconstruct images effectively based on top  $K$  eigen-vectors for the warping (say  $k_g=10$ ) and appearance (say  $k_a=50$ ). For the training images, we first align the images by warping their landmarks into the mean position (interpolation between landmarks is needed), and then compute the eigen-faces (appearance) from these aligned images. For each testing face: (i) project its landmarks to the top 10 eigen-warplings, we get the reconstructed landmarks, while losing a bit of geometric precision of reconstruction. (ii) warp the face image to the mean position and then project to the top  $k$  (say  $k=50$ ) eigen-faces, we get the reconstructed image at mean position, while further losing a bit of appearance accuracy. (iii) Warp the reconstructed faces in step (ii) to the positions reconstructed in step (i). Each new image is constructed from 60 numbers. Figure 4.18 demonstrates the reconstructed faces against the original testing images.



Figure 4.17: Left: the original first 100 testing faces. Center: the illumination channel of the first 100 aligned test faces which warping the test faces to the mean position of landmarks. Right: The reconstruction faces that warping the aligned test faces to their original position of landmarks. The data set contains 1000 images from CelebA, with different colors, illuminations, identities, viewing angles, shapes, and expressions. We divide the 1000 faces into two parts: the first 800 faces are put in a training set, and the remaining 200 faces are put in a test set.

AAM can also synthesize random faces by a random sampling of the landmarks and a random sampling of the appearance. Specifically, the eigen-faces and eigen-warplings for the landmarks can be seen as the appearance basis functions and the geometric basis functions for the generated faces, while the corresponding coefficients (latent vector) are computed by sampling from the eigen-axes, each axis has its own unit, i.e., the square-root of its eigen-value. As we can observe from Figure 4.18, the appearance basis functions usually captures the appearance information, such as illumination, color, and gender, while the geometric basis functions usually captures the geometric information, such as viewing angles and shape.

## 4.4 The Sparse FRAME Model

The FRAME model is a Markov random field model [17,71]. Originated from statistical physics, the Markov random field models or the Gibbs distributions are an important class of probability models for spatial processes such as those observed in natural images. The log probability density function of a Markov random

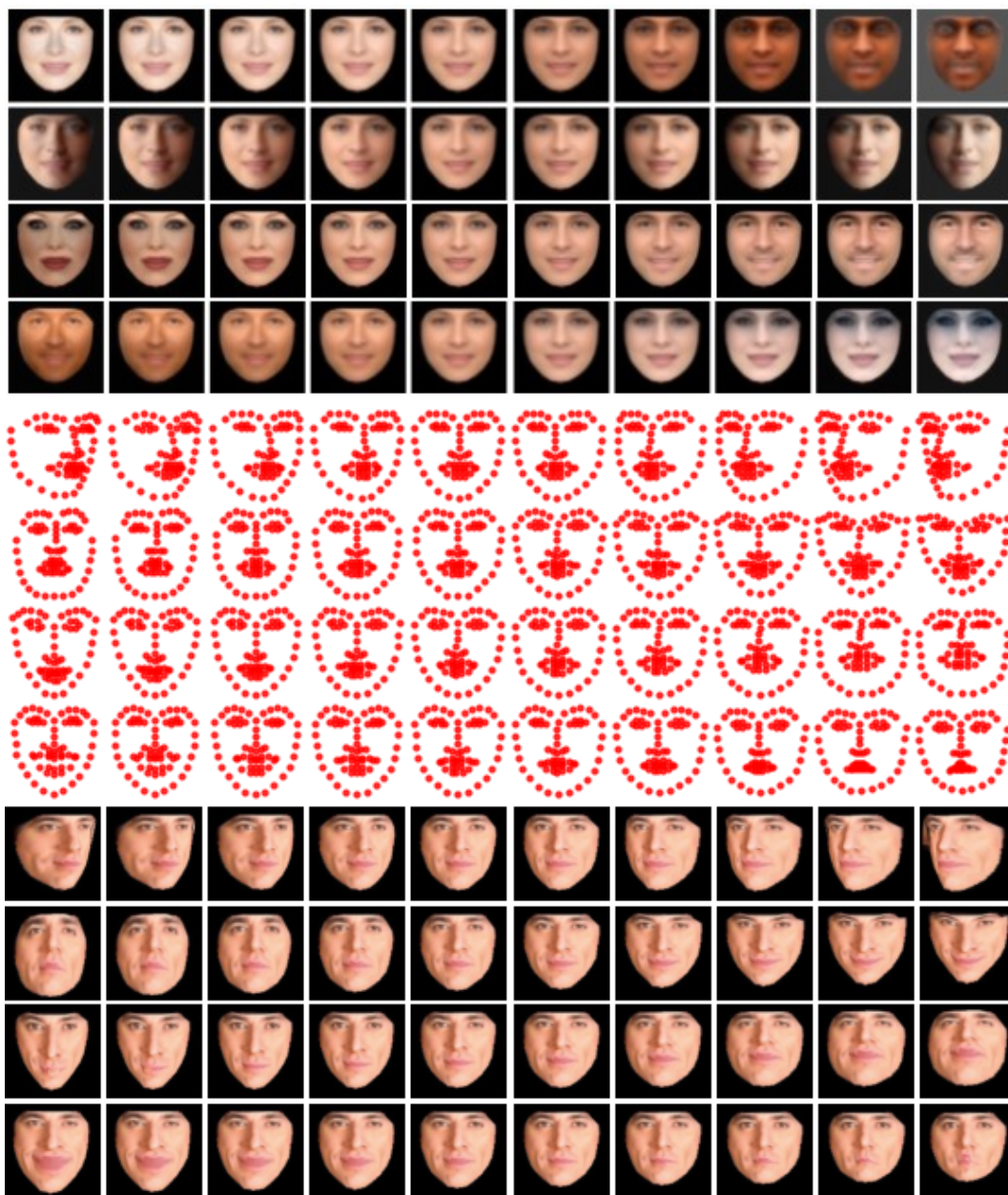


Figure 4.18: Top: Typical appearance basis functions, which is generated by setting the eigen-axes for the eigen-warpings of the landmarks to zero, and each time vary one dimension of the eigen-axes of the eigen-appearance-faces from  $[-\gamma \times std, \gamma \times std]$  with a uniform step  $\frac{2\gamma \times std}{10}$ , while holding the other eigen-axes of the eigen-appearance-faces at zero. Middle: Typical basis functions for the landmarks. Each row contains generated landmarks by sampling the eigen-value over one eigen-axis for the eigen-warpings, while keeping the coefficients over the other eigen-axes to be zeros. Down: Typical geometric basis functions, which is generated by warping the face of one people through the geometric basis of the landmarks from the Middle row.

field model is the sum of potential functions defined on the so-called cliques that consist of neighboring sites or pixels. The potential functions can be high-dimensional for those cliques that consist of many pixels, and it is difficult to learn such high-dimensional potential functions from the data. The FRAME model solves this problem by recruiting a bank of linear filters, and parametrizes the potential functions as point-wise one-dimensional non-linear transformations of linear filter responses. This model is the maximum entropy distribution that reproduces the marginal statistics such as marginal histograms of the filter responses, where for each filter, the marginal histogram is pooled over all the pixels in the image domain. The bank of filters can be designed, such as Gabor filters or Gabor wavelets tuned to different locations, scales and orientations. They can also be learned, together with the non-linear transformations, from the training data.

The FRAME model is originally developed for modeling spatially stationary processes such as stochastic textures, where the potential functions are translation invariant. One generalization of the FRAME model is a sparse FRAME model [239, 243] where the potential functions are location specific, and they are non-zero only at selected locations. This model is intended to model image patterns that are non-stationary in the spatial domain, such as object patterns. The model can be written as a shared sparse coding model, where the observed images are represented by a commonly shared set of wavelets selected from a dictionary. In this shared sparse coding model, the original linear filters for bottom-up computation (from image to filter responses) become linear basis functions for top-down representation (from coefficients to image).

#### 4.4.1 Dense FRAME

We start from the non-stationary or spatially inhomogeneous FRAME model [232, 239, 243] based on a dictionary of basis functions or wavelets  $\{B_{k,x}, \forall k, x\}$  (we assume that the dictionary of wavelets, such as the Gabor and DoG wavelets, has been given or has been learned by sparse component analysis [21, 157, 174]). The model is a random field of the following form:

$$p(\mathbf{I}; w) = \frac{1}{Z(w)} \exp \left[ \sum_{k=1}^K \sum_{x \in \mathcal{D}} w_{k,x} h(\langle \mathbf{I}, B_{k,x} \rangle) \right] q(\mathbf{I}). \quad (4.18)$$

The above model is a simple generalization of FRAME model, where  $\langle \mathbf{I}, B_{k,x} \rangle$  is the filter response, which can also be written as  $[F_k * \mathbf{I}](x)$ . The parameter  $w_{k,x}$  depends on position  $x$ , so the model is non-stationary.  $w = (w_{k,x}, \forall k, x)$ . Again  $Z(w)$  is the normalizing constant.  $h(\cdot)$  is a pre-specified rectification function. In [239],  $h(r) = |r|$ , i.e., the model is insensitive to the signs of filter responses.  $q(\mathbf{I})$  is a reference distribution, such as the Gaussian white noise model

$$q(\mathbf{I}) = \frac{1}{(2\pi\sigma^2)^{D/2}} \exp \left[ -\frac{1}{2\sigma^2} \|\mathbf{I}\|^2 \right], \quad (4.19)$$

where again  $D$  counts the number of pixels in the image domain  $\mathcal{D}$ .

#### 4.4.2 Sparse Representation

Assume we are given a dictionary of wavelets or basis functions  $\{B_{k,x}\}$ , where  $k$  may index a finite collection of prototype functions  $\{B_k, k = 1, \dots, K\}$ , and where  $B_{k,x}$  is a spatially translated copy of  $B_k$  to position  $x$ . We can represent an image  $\mathbf{I}$  by

$$\mathbf{I} = \sum_{k,x} c_{k,x} B_{k,x} + \epsilon, \quad (4.20)$$

where  $c_{k,x}$  are the coefficients, and  $\epsilon$  is the residual image. It is often assumed that the representation is sparse, i.e., most of the  $c_{k,x}$  are equal to zero. The resulting representation is also called sparse coding [47, 174].

The sparsification of  $c_{k,x}$ , i.e., the selection of the basis functions, can be accomplished by matching pursuit [152] or basis pursuit/Lasso [25, 210]. Using a Lasso-like objective function, the dictionary of basis functions  $\{B_k\}$  can be learned from a collection of training images [157, 174]. It is sometimes called sparse component analysis [43]. It can be considered a generalization of factor analysis. For natural images, the basis functions learned resemble the Gabor and DoG wavelets.

#### 4.4.3 Maximum Likelihood Learning

The basic learning algorithm estimates the parameters  $w = (w_{k,x}, \forall k, x)$  from a set of aligned training images  $\{\mathbf{I}_i, i = 1, \dots, n\}$  that come from the same category, where  $n$  is the total number of training images. The algorithm can be extended to learn from non-aligned images from mixed categories. The basic learning algorithm seeks to maximize the log-likelihood

$$L(w) = \frac{1}{n} \sum_{i=1}^n \log p(\mathbf{I}_i; w), \quad (4.21)$$

whose partial derivatives are

$$\frac{\partial L(w)}{\partial w_{k,x}} = \frac{1}{n} \sum_{i=1}^n h(\langle \mathbf{I}_i, B_{k,x} \rangle) - E_w [h(\langle \mathbf{I}, B_{k,x} \rangle)], \quad (4.22)$$

where  $E_w$  denotes expectation with respect to  $p(\mathbf{I}; w)$  in (4.18). This expectation can be approximated by Monte Carlo integration. Thus,  $w$  can be computed by the stochastic gradient ascent algorithm [186, 254]

$$w_{k,x}^{(t+1)} = w_{k,x}^{(t)} + \gamma_t \left[ \frac{1}{n} \sum_{i=1}^n h(\langle \mathbf{I}_i, B_{k,x} \rangle) - \frac{1}{\tilde{n}} \sum_{i=1}^{\tilde{n}} h(\langle \tilde{\mathbf{I}}_i, B_{k,x} \rangle) \right], \quad (4.23)$$

where  $\gamma_t$  is the step size or the learning rate, and  $\{\tilde{\mathbf{I}}_i, i = 1, \dots, \tilde{n}\}$  are the synthetic images sampled from  $p(\mathbf{I}; w^{(t)})$  using MCMC, such as Hamiltonian Monte Carlo [164] or the Gibbs sampler [70].  $\tilde{n}$  is the total number of independent parallel Markov chains that sample from  $p(\mathbf{I}; w^{(t)})$ .

#### 4.4.4 Generative Boosting

Model (4.18) is a dense model in that all the wavelets (or filters) in the dictionary are included in the model. We can sparsify the model by forcing most of the  $w_{k,x}$  to be zero, so that only a small number of wavelets are included in the model. This can be achieved by a generative version [243] of the epsilon-boosting algorithm [59, 62] (see also [35, 54, 228, 235]). The algorithm starts from  $w = 0$ , the zero vector. At the  $t$ -th iteration, let

$$\Delta_{k,x} = \frac{1}{n} \sum_{i=1}^n h(\langle \mathbf{I}_i, B_{k,x} \rangle) - \frac{1}{\tilde{n}} \sum_{i=1}^{\tilde{n}} h(\langle \tilde{\mathbf{I}}_i, B_{k,x} \rangle) \quad (4.24)$$

be the Monte Carlo estimate of  $\partial L(w)/\partial w_{k,x}$ , where again  $\{\tilde{\mathbf{I}}_i, i = 1, \dots, \tilde{n}\}$  are the synthetic images sampled from the current model. We select

$$(\hat{k}, \hat{x}) = \arg \max_{k,x} |\Delta_{k,x}|, \quad (4.25)$$



and update  $w_{\hat{k},\hat{x}}$  by

$$w_{\hat{k},\hat{x}} \leftarrow w_{\hat{k},\hat{x}} + \gamma_t \Delta_{\hat{k},\hat{x}}, \quad (4.26)$$

where  $\gamma_t$  is the step size, assumed to be sufficiently small (thus the term “epsilon” in the epsilon-boosting algorithm). We call this algorithm generative epsilon boosting because the derivatives are estimated by images generated from the current model. See Figure 4.19 for an illustration.

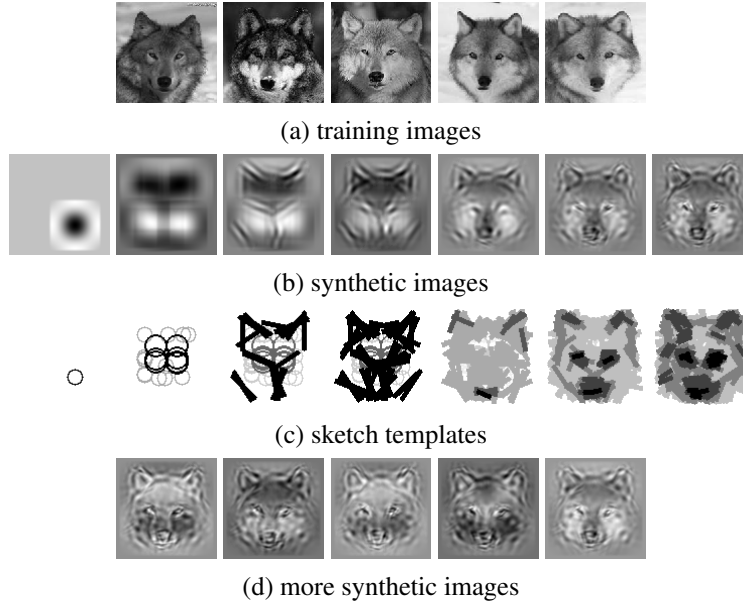


Figure 4.19: Reprinted with permission from [234]. Learning process of the generative boosting. (a) observed training images ( $100 \times 100$  pixels) from which the random field model is learned. (b) a sequence of synthetic images generated by the learned model as more and more wavelets are induced into the model. The numbers of the selected wavelets are 1, 20, 65, 100, 200, 500, and 800 respectively. (c) a sequence of sketch templates that illustrate the wavelets selected from the given dictionary. The dictionary includes 4 scales of Gabor wavelets, illustrated by bars of different sizes, and 2 scales of Difference of Gaussian (DoG) wavelets, illustrated by circles. In each template, smaller scale wavelets appear darker than larger ones. (d) more synthetic images independently generated from the final learned model.

The selected wavelet  $B_{\hat{k},\hat{x}}$  reveals the dimension along which the current model is most conspicuously lacking in reproducing the statistical properties of the training images. By including  $B_{\hat{k},\hat{x}}$  into the model and updating the corresponding parameter  $w_{\hat{k},\hat{x}}$ , the model receives the most needed boost. The process is like an artist making a painting, where  $B_{\hat{k},\hat{x}}$  is the stroke that is most needed to make the painting look more similar to the observed objects.

The epsilon boosting algorithm [62, 87] has an interesting relationship with the  $\ell_1$  regularization in the Lasso [210] and basis pursuit [25]. As pointed out by [187], under a monotonicity condition (e.g., the components of  $w$  keep increasing), such an algorithm approximately traces the solution path of the  $\ell_1$  regularized minimization of

$$-L(w) + \rho \|w\|_{\ell_1}, \quad (4.27)$$

where the regularization parameter  $\rho$  starts from a big value so that all the components of  $w$  are zero, and

gradually lowers itself to allow more components to be non-zero so that more wavelets are induced into the model.

#### 4.4.5 Sparse Model

After selecting  $m$  wavelets, we have the following sparse FRAME model:

$$p(\mathbf{I}; \mathbf{B}, w) = \frac{1}{Z(w)} \exp \left[ \sum_{j=1}^m w_j h(\langle \mathbf{I}, B_{k_j, x_j} \rangle) \right] q(\mathbf{I}), \quad (4.28)$$

where  $\mathbf{B} = (B_j = B_{k_j, x_j}, j = 1, \dots, m)$  is the set of wavelets selected from the dictionary, and  $w_j = w_{k_j, x_j}$ .

In model (4.28),  $m$  is much smaller than  $D$ , the number of pixels. Thus, we can represent  $\mathbf{I}$  by

$$\mathbf{I} = \sum_{j=1}^m c_j B_{k_j, x_j} + \epsilon, \quad (4.29)$$

where  $C = (c_j, j = 1, \dots, m)^\top$  are the least square regression coefficients of  $\mathbf{I}$  on  $\mathbf{B} = (B_j, j = 1, \dots, m)$ , i.e.,  $C = (\mathbf{B}^\top \mathbf{B})^{-1} \mathbf{B}^\top \mathbf{I}$ , and  $\epsilon$  is the residual image. The distribution of  $C$  under  $p(\mathbf{I}; \mathbf{B}, w)$  is

$$p_C(C; w) = \frac{1}{Z(w)} \exp \left[ \langle w, h(\mathbf{B}^\top \mathbf{B} C) \rangle \right] q_C(C), \quad (4.30)$$

where  $q_C(C)$  is the distribution of  $C$  under  $q(\mathbf{I})$ , and the transformation  $h(\cdot)$  is applied element-wise. Thus,  $p(\mathbf{I}; B, w)$  in (4.28) can be written as a wavelet sparse coding model (4.29) and (4.30). The forms of (4.28) and (4.29) show that the selected wavelets  $\{B_j\}$  serve both as filters and basis functions. The sparse coding form of the model (4.29) and (4.30) is used for sampling  $\{\tilde{\mathbf{I}}_i\}$  from  $p(\mathbf{I}; \mathbf{B}, w)$  by first sampling  $C \sim p_C(C; w)$  using the Gibbs sampler [70], and then generating  $\tilde{\mathbf{I}}_i$  according to (4.29).

Model (4.29) suggests that we can also select the wavelets by minimizing

$$\sum_{i=1}^n \left\| \mathbf{I}_i - \sum_{j=1}^m c_{i,j} B_{k_j, x_j} \right\|^2, \quad (4.31)$$

using a shared matching pursuit method [239]. See Figure 4.20 for an illustration. We can also allow the selected wavelets to perturb their locations and orientations to account for deformations [232].

The sparse FRAME model can be used for unsupervised learning tasks such as model-based clustering [52]. Extending the learning algorithm, one can learn a codebook of multiple sparse FRAME models from non-aligned images. The learned models can be used for tasks such as transfer learning [96, 239].

The sparse FRAME model merges two important research themes in image representation and modeling, namely, Markov random fields [17, 71] and wavelet sparse coding [157, 174].

The wavelets can be mapped to the first layer filters of a ConvNet [127] to be described below. The sparse FRAME models can be mapped to the second layer nodes of a ConvNet, except that the sparse FRAME versions of the second layer nodes are selectively and sparsely connected to the first layer nodes.

## 4.5 Compositional Sparse Coding

### 4.5.1 Sparsity and Composition

The goal of this section is to develop a compositional sparse code for natural images. Figure 4.21 illustrates the basic idea. We start with a dictionary of Gabor wavelets centered at a dense collection of locations



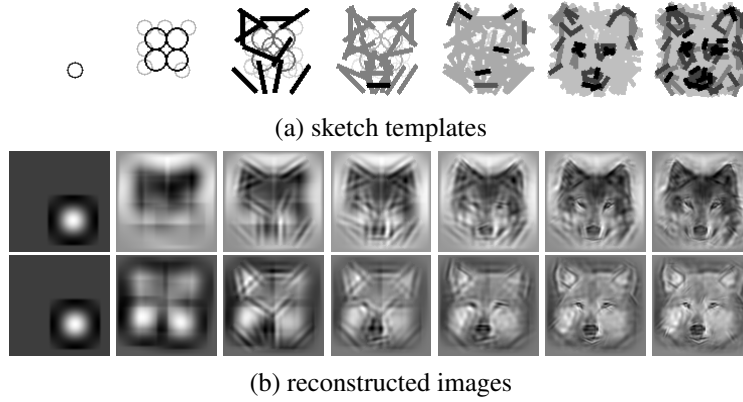


Figure 4.20: Reprinted with permission from [234]. Shared matching pursuit for the purpose of wavelet selection. (a) sequence of sketch templates that illustrate the wavelets selected sequentially in order to reconstruct all the training images simultaneously. The selected wavelets are shared by all the training images ( $100 \times 100$ ) in their reconstructions. The numbers of selected wavelets in the sequence are 2, 20, 60, 100, 200, 500, and 800 respectively. (b) sequences of reconstructed images by the selected wavelets for the 1st and 3rd training images in Figure 4.19(a).

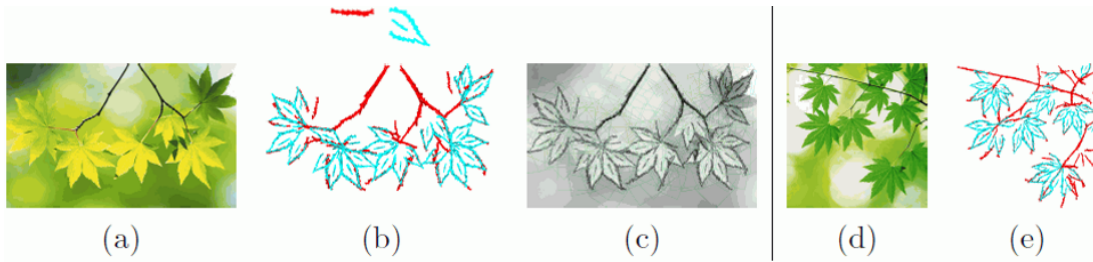


Figure 4.21: Reprinted with permission from [96]. Unsupervised learning of compositional sparse code (a,b,c) and using it for recognition and segmentation (d,e). (a) Training image of  $480 \times 768$  pixels. (b) Above: 2 compositional patterns (twig and leaf) in the form of shape templates learned from the training image. Each constituent Gabor wavelet (basis function) of a template is illustrated by a bar at the same location and with the same orientation and length as the corresponding wavelet. The size of each template is  $100 \times 100$  pixels. The number of basis functions in each template is no more than 40 and is automatically determined. Below: Representing the training image by translated, rotated, scaled and deformed copies of the 2 templates. (c) Superposing the deformed templates on the original image. The green squared boxes are the bounding boxes of the templates. (d) Testing image. (e) Representation (recognition) of the testing image by the 2 templates.

and tuned to a collection of scales and orientations. In Figure 4.21, each Gabor wavelet is illustrated by a bar at the same location and with the same length and orientation as the corresponding wavelet. Figure 4.21.(a) displays the training image. (b) displays a mini-dictionary of 2 compositional patterns of wavelets learned from the training image. Each compositional pattern is a template formed by a group of a small number of wavelets at selected locations and orientations. The learning is unsupervised in the sense that the images are not labeled or annotated. The number of templates in the dictionary is automatically determined by an adjusted Bayesian information criterion. The 2 templates are displayed in different colors, so that it

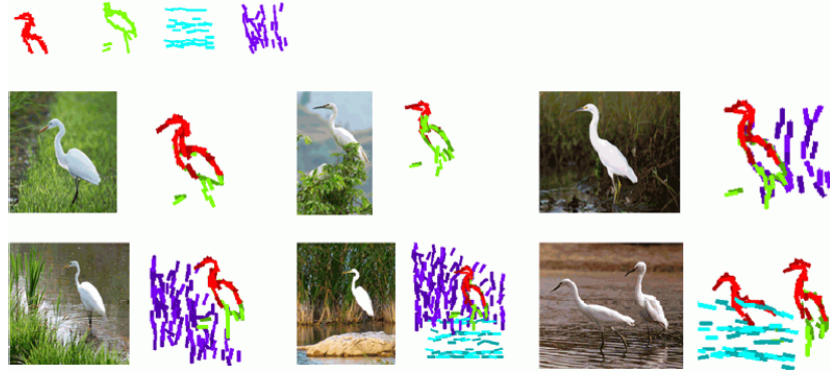


Figure 4.22: FReprinted with permission from [96]. our compositional patterns (templates) are learned from 20 training images (only 6 of them are shown in this figure). The training images are not registered or otherwise annotated. The size of each template is  $100 \times 100$  pixels. The number of basis functions in each template is no more than 40 and is automatically determined.

can be seen clearly how the translated, rotated, scaled and deformed copies of the 2 templates are used to represent the training image, as shown in (b). In (c), the templates are overlaid on the original image, where each green squared box is the bounding box of the template. In our current implementation, we allow some overlap between the bounding boxes of the templates. The templates learned from the training image can be generalized to testing images, as shown in (d) and (e).

Figure 4.22 shows another example, where part templates of egrets and templates of water waves and grasses are learned from 20 training images without supervision. That is, the training images are not registered, in that we do not assume that the objects in the training images appear at the same location and scale. It is interesting to observe that in this example, the unsupervised learning also accomplishes image segmentation, object detection and perceptual grouping (e.g., grass pattern), which are important tasks in vision.

Our compositional sparse code combines two fundamental principles in image representation and computational vision, namely, *sparsity* and *compositionality*. We shall briefly review these two principles below and then give an overview of our methodology.

The compositionality principle was proposed in the context of computer vision by Geman, Potter, and Chi (1998) and Zhu and Mumford (2007). The principle holds that patterns in natural images are compositions of parts, which are in turn compositions of sub-parts, and so on. An interesting example cited by Geman et al. is Laplace’s remark that one strongly prefers to view the string CONSTANTINOPLE as a single word, rather than 14 individual letters. This is also the case with the basis functions in the sparse coding of natural images. Like letters forming the words, the basis functions in the sparse representations of natural images also form various compositional patterns in terms of their spatial arrangements. We call such sparsity the compositional sparsity, which is a special form of structured sparsity.

Structured sparsity has received considerable attention in statistics and machine learning in recent years. The most prominent example is the group Lasso, which replaces the  $\ell_1$  penalty of Lasso by a composite penalty based on the group structure among the basis functions. In the group Lasso, the collection of the groups is assumed given. In our work, however, we do not assume that the groups are given, and we seek to learn dictionaries of the recurring compositional patterns in the spatial grouping of the basis functions.

Any hierarchical compositional model will necessarily end with constituent elements that cannot be further decomposed, and such elements may be called “atoms.” Interestingly, the basis functions are commonly referred to as atoms in sparse coding literature, and the sparse representation based on atoms is usually called

“atomic decomposition” . Compositionality enables us to compose atoms into composite representational units, which leads to much sparser and thus more meaningful representations of the signals.

The current form of our model consists of two layers of representational units: basis functions and shape templates. It is possible to extend it to multiple layers of hierarchy.

#### 4.5.2 Compositional Sparse Coding Model

We strive to write down our model in an analogous form as the Olshausen-Field model  $\mathbf{I}_m = \sum_{i=1}^n c_{m,i} B_{x_m,i,s_m,i,\alpha_{m,i}} + U_m$ , by making the notation compact.

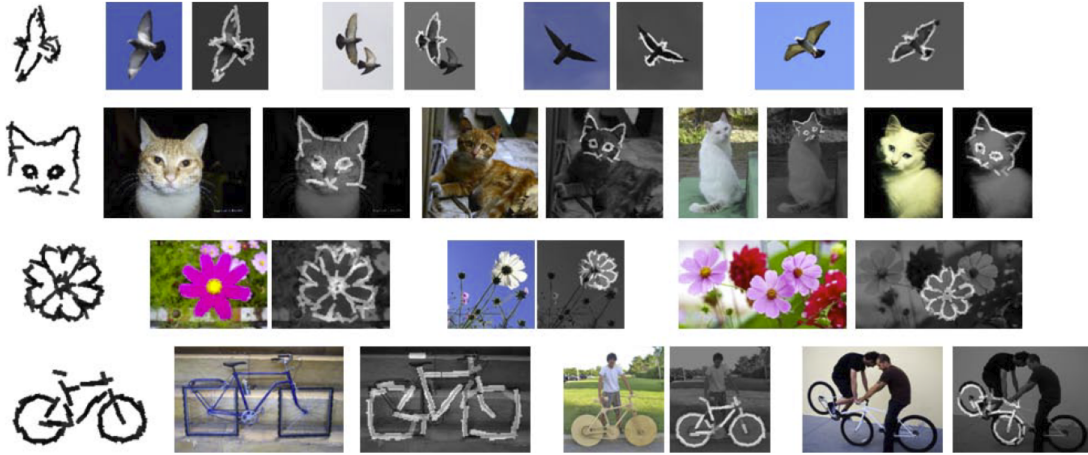


Figure 4.23: Reprinted with permission from [96]. Objects appear at different locations, scales and orientations in the training images. In each row, the first plot displays the nominal active basis template. The rest of the row displays some examples of training images and the suppositions of the spatially translated, scaled, rotated and deformed versions of the nominal template.

*Compact notation.* As the first step, let us slightly generalize the active basis model by assuming that the template may appear at location  $X_m$  in image  $\mathbf{I}_m$ , then we can write the representation in the following form:

$$\begin{aligned} \mathbf{I}_m &= \sum_{i=1}^n c_{m,i} B_{X_m+x_i+\Delta x_{m,i},s,\alpha_i+\Delta\alpha_{m,i}} + U_m \\ &= C_m \mathbf{B}_{X_m} + U_m, \end{aligned} \quad (4.32)$$

where  $\mathbf{B}_{X_m} = (B_{X_m+x_i+\Delta x_{m,i},s,\alpha_i+\Delta\alpha_{m,i}}, i = 1, \dots, n)$  is the deformed template spatially translated to  $X_m$ ,  $C_m = (c_{m,i}, i = 1, \dots, n)$ , and  $C_m \mathbf{B}_{X_m} = \sum_{i=1}^n c_{m,i} B_{X_m+x_i+\Delta x_{m,i},s,\alpha_i+\Delta\alpha_{m,i}}$  by definition.

$\mathbf{B}_{X_m}$  explains the part of  $\mathbf{I}_m$  that is covered by  $\mathbf{B}_{X_m}$ . For each image  $\mathbf{I}_m$  and each  $X_m$ , we can define the log-likelihood ratio similar to (4.17):

$$\begin{aligned} l(\mathbf{I}_m | \mathbf{B}_{X_m}) &= \log \frac{p(\mathbf{I}_m | \mathbf{B}_{X_m})}{q(\mathbf{I}_m)} \\ &= \sum_{i=1}^n \left[ \lambda_i \max_{\Delta x, \Delta \alpha} h(|\langle \mathbf{I}_m, B_{X_m+x_i+\Delta x,s,\alpha_i+\Delta\alpha} \rangle|^2) - \log Z(\lambda_i) \right]. \end{aligned} \quad (4.33)$$

As the next step of this modeling procedure, in addition to spatial translation and deformation, we can also rotate and scale the template. So a more general version of (4.32) is

$$\mathbf{I}_m = C_m \mathbf{B}_{X_m, S_m, A_m} + U_m, \quad (4.34)$$

where  $X_m$  is the location,  $S_m$  is the scale, and  $A_m$  is the orientation of the translated, rotated, scaled and deformed template. The scaling of the template is implemented by changing the resolution of the original image. We adopt the convention that whenever the notation  $\mathbf{B}$  appears in image representation, it always means the deformed template, where the perturbations of the basis functions can be inferred by local max pooling. The log-likelihood ratio  $l(\mathbf{I}_m \mid \mathbf{B}_{X_m, S_m, A_m})$  can be similarly defined as in (4.33). Figure 4.23 illustrates the basic idea of representation (4.34). In addition to spatial translation, dilation and rotation of the template, we may also allow mirror reflection as well as the change of aspect ratio.

*Compact representation.* Now suppose we have a dictionary of  $T$  active basis templates,  $\{\mathbf{B}^{(t)}, t = 1, \dots, T\}$ , where each  $\mathbf{B}^{(t)}$  is a compositional pattern of basis functions. Then we can represent the image  $\mathbf{I}_m$  by  $K_m$  templates that are spatially translated, rotated, scaled and deformed versions of these  $T$  templates in the dictionary:

$$\mathbf{I}_m = \sum_{k=1}^{K_m} C_{m,k} \mathbf{B}_{X_{m,k}, S_{m,k}, A_{m,k}}^{(t_{m,k})} + U_m, \quad (4.35)$$

where each  $\mathbf{B}_{X_{m,k}, S_{m,k}, A_{m,k}}^{(t_{m,k})}$  is obtained by translating the template of type  $t_{m,k}$ , i.e.,  $\mathbf{B}^{(t_{m,k})}$ , to location  $X_{m,k}$ , scaling it to scale  $S_{m,k}$ , rotating it to orientation  $A_{m,k}$ , and deforming it to match  $\mathbf{I}_m$ . Note that according to (4.35), the images represented by the dictionary are no longer assumed to be aligned.

If the  $K_m$  templates do not overlap with each other, then the log-likelihood ratio is

$$\sum_{m=1}^M \sum_{k=1}^{K_m} \left[ l(\mathbf{I}_m \mid \mathbf{B}_{X_{m,k}, S_{m,k}, A_{m,k}}^{(t_{m,k})}) \right]. \quad (4.36)$$

*Packing and unpacking.* The above representation is in analogy to model (4.9) in subsection (4.3.1), which we copy here:  $\mathbf{I}_m = \sum_{i=1}^n c_{m,i} B_{x_{m,i}, s_{m,i}, \alpha_{m,i}} + U_m$ . The difference is that each  $\mathbf{B}_{X_{m,k}, S_{m,k}, A_{m,k}}^{(t_{m,k})}$  is a composite representational unit, which is itself a group of basis functions that follow a certain compositional pattern of type  $t_{m,k}$ . Because of such grouping or packing, the number of templates  $K_m$  needed to encode  $\mathbf{I}_m$  is expected to be much smaller than the total number of basis functions needed to represent  $\mathbf{I}_m$ , thus resulting in sparser representation. Specifically, if each template is a group of  $g$  basis functions, then the number of basis functions in the representation (4.35) is  $K_m g$ . In fact, we can unpack model (4.35) into the representation (4.9). The reason that it is advantageous to pack the basis functions into groups is that these groups exhibit  $T$  types of frequently occurring spatial grouping patterns, so that when we encode the image  $\mathbf{I}_m$ , for each selected group  $\mathbf{B}_{X_{m,k}, S_{m,k}, A_{m,k}}^{(t_{m,k})}$ , we only need to code the overall location, scale, orientation and type of the group, instead of the locations, scales and orientations of the individual constituent basis functions.

*Limited overlap assumption.* It is desirable to allow some limited overlap between the bounding boxes of the  $K_m$  templates that encode  $\mathbf{I}_m$ . Even if the bounding boxes of two templates have some overlap with each other, their constituent basis functions may not overlap much. If we do not allow any overlap between the bounding boxes of the templates, some salient structures of  $\mathbf{I}_m$  may fall through the cracks between the templates. Also, it is possible that the frequently occurring patterns may actually overlap with each other. For instance, in a string “ABABABA”, the pattern “AB” is frequently occurring, but at the same time, the

pattern “BA” is as frequent as “AB”, and these two patterns overlap with each other. So it can be desirable to allow some overlap between the patterns in order to recover all the important recurring patterns. On the other hand, we do not want to allow excessive overlap between the templates. Otherwise the learned templates will be too redundant, and we will need a lot of them in order to describe the training images. In practice, we assume the following limited overlap constraint: For each template  $\mathbf{B}_{X_{m,k}, S_{m,k}, A_{m,k}}^{(t_{m,k})}$  centered at  $X_{m,k}$ , let  $D$  be the side length of its squared bounding box, then no other templates are allowed to be centered within a distance of  $\rho D$  from  $X_{m,k}$  (default setting:  $\rho = .4$ ).

Such an assumption naturally leads to an inhibition step when we use a dictionary of templates to encode a training or testing image. Specifically, when a template is chosen to encode an image, this template will prevent overlapping templates from being selected. The template matching pursuit algorithm to be described below adopts such an inhibition scheme.

The following are the details of the two steps.

*Step (I): Image encoding by template matching pursuit.* Suppose we are given the current dictionary  $\{\mathbf{B}^{(t)}, t = 1, \dots, T\}$ . Then for each  $\mathbf{I}_m$ , the template matching pursuit process seeks to represent  $\mathbf{I}_m$  by sequentially selecting a small number of templates from the dictionary. Each selection seeks to maximally increase the log-likelihood ratio (4.36).

[I.0] Initialize the maps of template matching scores for all  $(X, S, A, t)$ :

$$\mathbf{R}_m^{(t)}(X, S, A) \leftarrow l(\mathbf{I}_m | \mathbf{B}_{X,S,A}^{(t)}) - n^{(t)}\gamma,$$

where  $n^{(t)}$  is the number of basis functions in the  $t$ -th template in the dictionary and  $\gamma$  is a constant controlling model complexity as explained above. This can be accomplished by first rotating the template  $\mathbf{B}^{(t)}$  to orientation  $A$ , and then scanning the rotated template over the image zoomed to the resolution that corresponds to scale  $S$ . The larger the  $S$  is, the smaller the resolution is. Initialize  $k \leftarrow 1$ .

[I.1] Select the translated, rotated, scaled and deformed template by finding the global maximum of the response maps:

$$(X_{m,k}, S_{m,k}, A_{m,k}, t_{m,k}) = \arg \max_{X,S,A,t} \mathbf{R}_m^{(t)}(X, S, A).$$

[I.2] Let the selected arg-max template inhibit overlapping candidate templates to enforce limited overlap constraint. Let  $D$  be the side length of the bounding box of the selected template  $\mathbf{B}_{X_{m,k}, S_{m,k}, A_{m,k}}^{(t_{m,k})}$ , then for all  $(X, S, A, t)$ , if  $X$  is within a distance  $\rho D$  from  $X_{m,k}$ , then set the response  $\mathbf{R}_m^{(t)}(X, S, A) \leftarrow -\infty$  (default setting:  $\rho = .4$ ).

[I.3] Stop if all  $\mathbf{R}_m^{(t)}(X, S, A, t) < 0$ . Otherwise let  $k \leftarrow k + 1$ , and go to [I.1].

The template matching pursuit algorithm implements a hard inhibition to enforce the limited overlap constraint. In a more rigorous implementation, we may update the residual image by  $U_m \leftarrow U_m - C_m \mathbf{B}_{X_{m,k}, S_{m,k}, A_{m,k}}^{(t_{m,k})}$  as in the original version of matching pursuit. But the current simplified version is more efficient.

*Step (II): Dictionary re-learning by shared matching pursuit.* For each  $t = 1, \dots, T$ , we re-learn  $\mathbf{B}^{(t)}$  from all the image patches that are currently covered by  $\mathbf{B}^{(t)}$ . Each iteration of the shared matching pursuit process seeks to maximally increase the penalized log-likelihood ratio (??), given the current encoding  $(t_{m,k}, X_{m,k}, S_{m,k}, A_{m,k}, k = 1, \dots, K_m)$ .

[II.0] Image patch cropping. For each  $\mathbf{I}_m$ , go through all the selected templates  $\{\mathbf{B}_{X_{m,k}, S_{m,k}, A_{m,k}}^{(t_{m,k})}, \forall k\}$  that encode  $\mathbf{I}_m$ . If  $t_{m,k} = t$ , then crop the image patch of  $\mathbf{I}_m$  (at the resolution that corresponds to  $S_{m,k}$ ) covered by the bounding box of the template  $\mathbf{B}_{X_{m,k}, S_{m,k}, A_{m,k}}^{(t_{m,k})}$ .

[II.1] Template re-learning. Re-learn template  $\mathbf{B}^{(t)}$  from all the image patches covered by  $\mathbf{B}^{(t)}$  that are cropped in [II.0], with their bounding boxes aligned. The learning is accomplished by the shared matching pursuit algorithm of subsection (4.3.5).

This dictionary re-learning step re-learns each compositional pattern from the re-aligned raw image patches, where the sparse representations and the correspondences between the selected basis functions are obtained simultaneously by the shared matching pursuit algorithm, thus avoiding the difficulty faced by the sparsify-and-then-compose procedure of Strategy I.

*Random initialization and polarization.* As mentioned above, the learning algorithm is initialized by learning from image patches randomly cropped from the training images. As a result, the initially learned templates are rather meaningless, but meaningful templates emerge very quickly after a few iterations.

During the template matching pursuit process of the first iteration, we continue to select templates even if the template matching scores are below zero. In the first iteration, we stop the template matching pursuit process for each  $\mathbf{I}_m$  when all the candidate templates are inhibited on  $\mathbf{I}_m$ . That is because the initial templates in the dictionary are rather random, so we force them to explain the whole image of  $\mathbf{I}_m$  even if the initial templates do not match the image well.

In the beginning, the differences among the initial templates are small. However, as the algorithm proceeds, the small differences among the initial templates trigger a polarizing or specializing process, so that the templates become more and more different, and they specialize in encoding different types of image patches.

## 4.6 Bottom-up Filters or Top-down Basis Functions?

In the FRAME model, the Gabor wavelets serve as bottom-up filters to extract features. In the texton model or active basis model, the Gabor wavelets serve as top-down basis functions. It is still not entirely clear how to reconcile the two roles.

In terms of modeling and computing, the FRAME model based on bottom-up filters require MCMC such as Langevin dynamics to sample from it. The texton model or the active basis model based on top-down basis functions requires inference process to select and infer the coefficients of the basis functions.

For simplicity, we have assumed that the Gabor wavelets are pre-designed. They can also be learned from training data.

# 5

## Gestalt Laws and Perceptual Organization

### 5.1 Gestalt Laws for Perceptual Organization

The word Gestalt, as used in Gestalt Psychology is often understood as “pattern”. The school of Gestaltism, emerged in Austria and Germany in the early twentieth century, focuses on the study on how organisms perceive entire patterns or configurations from an image, rather than just individual components. As is commonly cited, “the whole is more than the sum of its parts” best describes the idea of Gestaltism.

Many people share the experience that when we look at the world, we tend to decompose complex scenes into groups of objects against a background, and perceive the objects as a composition of parts, and sometimes even those parts have sub-parts. But what actually enables us to do this? Does that surprise you that we manages to do such a remarkable achievement when what we see is, to some extend, just a distribution of colored points? The Gestalt Psychology believes that we are able to do it because our vision system favors a set of principles when we understand the world. The set of principles are latter formulated based on regularities based on wholes, sub-wholes, groups, or Gestalten, and called Gestalt laws. Gestalt laws, proximity, similarity, figure-ground, continuity, closure, and connection, determine how humans perceive visuals in connection with different objects and environments.

Specifically, Gestalt laws include the following:

- Law of Similarity: The law of similarity suggests that similar things tend to show up together. The grouping can occur in various modalities, including visual and auditory stimuli.
- Law of Pragnanz: “Pragnanz” in German means “good figure”. Therefore, this law is sometimes referred to as the law of good figure or the law of simplicity. It states that we perceive figures in the simplest way possible, say a composition of simple shapes.
- Law of Proximity: The law of proximity suggests that when we perceive an image, closer objects tend to be grouped together. This law could could be particularly helpful when we describe a set of objects and explain how we separate them into several smaller groups.
- Law of Continuity: We often have the impression that points connected by lines or curves form a smooth path together instead of segmented lines and angles. The Gestaltism explains this phenomenon using the law of continuity.
- Law of Closure: If things, when grouped as a whole, would become a simple entity, we usually ignore contradictory evidence and choose to fill in the missing pieces to treat it as a group. This law of closure helps us understand segmented arcs of a circle to be a whole.

- Law of Common Region: According to this law, elements in the same region of space tend to be grouped together.

Gestalt laws provide a way for us to understand some important perception heuristics and psychological research continues to offer insights into it. But from a modeling point of view, despite long standing observations in the psychology literature, there was no explicit mathematical models that can account for these Gestalt laws and weight them properly when multiple laws are working together or in competition.

## 5.2 Texton Process Embedding Gestalt Laws

In this section, we discuss a way to model visual patterns based on Gestalt laws by integrating descriptive and generative methods.

In particular, we present a mathematical framework for visual learning that integrates two popular statistical learning paradigms in the literature:

1. Descriptive methods, such as Markov random fields and minimax entropy learning [262]
2. Generative methods, such as principal component analysis, independent component analysis [13], and transformed component analysis [55], wavelet coding [25, 150], and sparse coding [136, 172].

The integrated framework creates richer classes of probabilistic models for visual patterns. In this section, we demonstrate the integrated framework by learning a class of hierarchical models for texton patterns (the term of textons was coined by psychologist Julesz in the early 80s). At the bottom level of the model, we assume that an observed texture image is generated by multiple hidden “texton maps”, and textons on each map are translated, scaled, and oriented versions of a window function, like mini-templates or wavelet bases. The texton maps generate the observed image by occlusion or linear superposition. This bottom level of the model is generative in nature. At the top level of the model, the spatial arrangements or global organizations of the textons in the texton maps are characterized by the minimax entropy principle, which leads to embellished versions of Gibbs point process models [26]. The top level of the model is descriptive in nature.

The learning framework achieves four goals:

1. Computing the window functions (or appearances) of different types of textons.
2. Inferring the hidden texton maps that generate the image.
3. Learning Gibbs point process models for the texton maps.
4. Verifying the learned window functions and Gibbs models through texture synthesis by stochastic sampling.

We use a stochastic gradient algorithm for inferential computation. We demonstrate the learning framework by a set of experiments.

### 5.2.1 Introduction

What a vision algorithm can accomplish depends crucially upon how much it “understands” the contents of the observed images. Thus in computer vision and especially Bayesian image analysis, an important research theme is visual learning whose objective is to construct parsimonious and general models that can



realistically characterize visual patterns in natural scenes. Due to the stochastic nature of visual patterns, visual learning is posed as a statistical modeling and inference problem, and existing methods in the literature can be generally divided into two categories. In this book, we call one category the *descriptive methods* and the other category the *generative methods*.\*

Descriptive methods model a visual pattern by imposing statistical constraints on features extracted from signals. Descriptive methods include Markov random fields, minimax entropy learning [262], deformable models, etc. For example, recent methods on texture modeling all fall into this category [19, 89, 180, 262]. These models are built on pixel intensities or some deterministic transforms of the original signals, such as linear filtering. The shortcomings of descriptive methods are two-fold. First, they do not capture high-level semantics in visual patterns, which are often very important in human perception. For example, a descriptive model of texture can realize a cheetah skin pattern with impressive synthesis results but it does not have an explicit notion of individual blobs. Second, as descriptive models are built directly on the original signals, the resulting probability densities are often of very high dimensions and the sampling and inference are computationally expensive. It is desirable to have dimension reduction so that the models can be built in a low dimensional space that often better reflects the intrinsic complexity of the pattern.

In contrast to descriptive methods, generative methods postulate hidden variables as the causes for the complicated dependencies in raw signals, and thus the models are hierarchical in nature. Generative methods are widely used in vision and image analysis. For example, principal component analysis (PCA), independent component analysis (ICA) [13], transformed component analysis (TCA) [55], wavelet image representation [25, 150], sparse coding [136, 172], and the random collage model for generic natural images [130]. Despite their simplicity, these generative models suffer from an over-simplified assumption that hidden variables are independent and identically distributed (iid).<sup>†</sup> As a result, they are not sophisticated enough to generate realistic visual patterns. For example, a wavelet image coding model can easily reconstruct an observed image, but it cannot synthesize a texture pattern through iid random sampling because the spatial relationships between the wavelet coefficients are not characterized.

The two learning paradigms were developed almost independently by somewhat disjoint communities working on different problems, and their relationship has yet to be studied. Therefore, we present a visual learning framework that integrates both descriptive and generative methods and extends them to a richer class of probabilistic models for computer vision.

The integrated learning framework makes contributions to visual learning in the following four aspects.



Figure 5.1: Two examples of natural patterns.

---

\*There is a third category of methods which are discriminative. The goal of discriminative methods is not for modeling visual patterns explicitly but for classification. For example, pattern recognition, feed-forward neural networks, and classification trees, etc. Thus we choose not to discuss it because our focus is statistical modeling.

<sup>†</sup>Interested readers are referred to a recent paper [189] for discussion of the problem with existing generative models.

First, it combines the advantages of both descriptive and generative methods, and provides a general visual learning framework for modeling complex visual patterns. In computer vision, a fundamental observation, stated in Marr’s primal sketch paradigm [153], is that natural visual patterns consist of multiple layers of stochastic processes. For example, Fig. 5.1 displays two natural images. When we look at the ivy-wall image, we perceive not only the texture “impression” in terms of pixel intensities, but we also see the repeated elements in the ivy and bricks. To capture the hierarchical notion, we propose a multi-layer generative model as shown in Fig 5.2. We assume that a texture image is generated by a few layers of stochastic processes and each layer consists of a finite number of distinct but similar elements, called “textons” (following the terminology of Julesz). In experiments, each texton covers more than 100 pixels on average, so the layered representation achieves a nearly 100-fold dimension reduction.<sup>‡</sup> The spatial arrangements of the textons at each layer are characterized by Markov random field (MRF) models through the minimax entropy learning [262], and previous MRF texture models can be considered special cases where the models have only one layer and each “texton” is just a pixel.

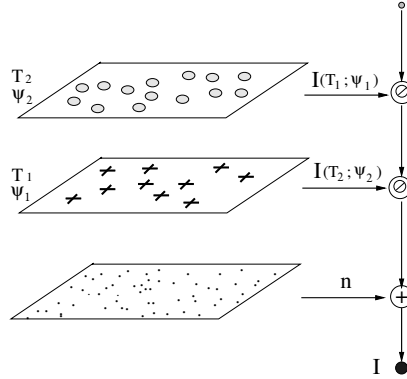


Figure 5.2: Reprinted with permission from [80]. A generative model for an image  $\mathbf{I}$  consists of multiple layers of texton maps  $\mathbf{I}(\mathbf{T}_l; \Psi_l)$ ,  $l = 1, \dots, L$  superimposed with occlusion plus an additive noise image  $\mathbf{n}$ .

It is our belief that *descriptive models are precursors of generative models* and both are ingredients of the integrated learning process. In visual learning, the model can be initially built on image intensities via some features computed deterministically from the image intensities. Then we can replace the features with hidden causes, and such a process would incrementally discover more abstract elements or concepts such as textons, curves, flows, and so on, where elements at the more abstract levels become causes for the elements of lower abstractions. For instance, the flows generate curves, and the curves generate textons, which in turn generate pixel intensities. At each stage, the elements at the most abstract level have no further hidden causes and thus they have to be characterized by a descriptive model based on some deterministic features, and such models can be derived by the minimax entropy principle as demonstrated in [233]. When a new hidden level of elements is introduced, it replaces the current descriptive model with a simplified one. The learning process evolves until the descriptive model for the most abstract elements becomes simple enough for a certain vision purpose. By analogy, the learning process is very similar to the situation in physics, where experimental observations are explained by a hierarchy of elements (say from quarks, electrons, atoms, to molecules) and their interactions.

Second, the integrated learning framework provides a representational definition of “textons”. Texton has been an important notion in texture perception and early vision. Unfortunately, it was only expressed vaguely in psychology [114], and a precise definition of texton has yet to be found. In this chapter, we

<sup>‡</sup>A texton has to be described by a few variables for location, scale, orientation, etc.

argue that the definition of “texton” is possible only in the context of a generative model. In contrast to the constraint based clustering method by Malik, Leung, etc. [134, 135, 147], in this book, textons are naturally embedded in a generative model and are inferred as hidden variables of the generative model. This is consistent with the philosophy of ICA [13], TCA [55] and sparse coding [136, 172].

Third, we present a *Gestalt ensemble* to characterize the hidden texton maps as attributed point processes. The Gestalt ensemble corresponds to the grand canonical ensemble in statistical physics [23], and it differs from traditional Gibbs models by having an unknown number of textons whose neighborhood changes dynamically. The relationships between neighboring textons are captured by some Gestalt laws, such as proximity and continuity, etc.

Fourth, we adapt a stochastic gradient algorithm [79] for effective learning and inference, in contrast to the conventional EM algorithm [36]. In the adapted algorithm, we simplify the original likelihood function and solve the simplified maximum likelihood problem first. Starting from the initial solution, we then use the stochastic gradient algorithm to find refined solutions.

We demonstrate the proposed learning method on texture images. For an input texture image, the learning algorithm achieves the following four objectives.

1. Learning the appearance of textons for each stochastic process. Textons of the same stochastic process are translated, scaled, and oriented versions of a window function, like mini-templates or wavelet bases.
2. Inferring the hidden texton maps, each of which consists of an unknown number of similar textons which are related to each other by affine transformations.
3. Learning the minimax entropy models for the stochastic processes that generate the textons maps.
4. Verifying the learned window functions and generative models through stochastic sampling.

It is worth noting that this is not another paper on texture synthesis, which is a computer graphics problem that has been successfully solved by a variety of texture sampling techniques [19, 89, 180, 262]. Instead, this is about learning parsimonious and sufficient models for vision tasks, where texture synthesis only serves as a method for model checking.

The section is organized as follows. Section (5.2.2) introduces the background on both generative and descriptive learning methods. Section (5.2.3) discusses a hierarchical model for texture. Section (5.2.4) studies Gestalt ensembles for modeling texton processes. Then section (5.2.5) presents an integrated learning method. Section (8.2.3) presents the algorithm for inferential computation. Some experiments are shown in Section (5.2.6). We conclude with a critical discussion in section (5.2.7).

## 5.2.2 Background on Descriptive and Generative Learning

Given a set of images  $\mathcal{I} = \{\mathbf{I}_1^{\text{obs}}, \dots, \mathbf{I}_M^{\text{obs}}\}$ , where  $\mathbf{I}_m^{\text{obs}}, m = 1, \dots, M$  are considered realizations of some underlying stochastic process governed by a frequency  $f(\mathbf{I})$ . The objective of visual learning is to estimate a probabilistic model  $p(\mathbf{I})$  based on  $\mathcal{I}$  so that  $p(\mathbf{I})$  approaches  $f(\mathbf{I})$  by minimizing a Kullback-Leibler divergence  $\text{KL}(f||p)$  from  $f$  to  $p$  [28],

$$\text{KL}(f||p) = \int f(\mathbf{I}) \log \frac{f(\mathbf{I})}{p(\mathbf{I})} d\mathbf{I} = E_f[\log f(\mathbf{I})] - E_f[\log p(\mathbf{I})]. \quad (5.1)$$

In practice, the expectation  $E_f[\log p(\mathbf{I})]$  is replaced by a sample average. Thus we have the standard maximum likelihood estimator (MLE),

$$p^* = \arg \min_{p \in \Omega_p} \text{KL}(f||p) \approx \arg \max_{p \in \Omega_p} \sum_{m=1}^M \log p(\mathbf{I}_m^{\text{obs}}), \quad (5.2)$$

where  $\Omega_p$  is the family of distributions where  $p^*$  is searched for. One general procedure is to search for  $p$  in a sequence of nested probability families,

$$\Omega_0 \subset \Omega_1 \subset \dots \subset \Omega_K \rightarrow \Omega_f \ni f,$$

where  $K$  indexes the dimensionality of the space. For example,  $K$  could be the number of free parameters in a model. As  $K$  increases, the probability family should be general enough to approach  $f$  to an arbitrary preset precision.

There are two choices of families for  $\Omega_p$  in the literature and both are general enough for approximating any distribution  $f$ .

*The first choice is the exponential family.* The exponential family can be derived by the descriptive method through maximum entropy, and has its root in statistical mechanics [23]. A descriptive method extracts a set of  $K$  feature statistics as *deterministic transforms* of an image  $\mathbf{I}$ , denoted by  $\phi_k(\mathbf{I}), k = 1, \dots, K$ . Then it constructs a model  $p$  by imposing *descriptive constraints* so that  $p$  reproduces the observed statistics  $\mathbf{h}_k^{\text{obs}}$  extracted from  $\mathcal{I}$ ,

$$E_p[\phi_k(\mathbf{I})] = \mathbf{h}_k^{\text{obs}} = \frac{1}{M} \sum_{m=1}^M \phi_k(\mathbf{I}_m^{\text{obs}}) \approx E_f[\phi_k(\mathbf{I})] = \mathbf{h}_k, \quad k = 1, \dots, K. \quad (5.3)$$

One may consider  $\mathbf{h}_k$  as a projected statistics of  $f(\mathbf{I})$ , thus when  $M$  is large enough,  $p$  and  $f$  will have the same projected (marginal) statistics on the  $K$  chosen dimensions. By the maximum entropy principle [110], this leads to the Gibbs model,

$$p(\mathbf{I}; \boldsymbol{\beta}) = \frac{1}{Z(\boldsymbol{\beta})} \exp\left\{-\sum_{k=1}^K \beta_k \phi_k(\mathbf{I})\right\}.$$

The parameters  $\boldsymbol{\beta} = (\beta_1, \dots, \beta_K)$  are Lagrange multipliers and they are computed by solving the constraint equations (5.3). The  $K$  features are chosen by a minimum entropy principle [262].

The descriptive learning method augments the dimension of the space  $\Omega_p$  by increasing the number of feature statistics and generates a sequence of exponential families,

$$\Omega_1^d \subset \Omega_2^d \subset \dots \Omega_K^d \rightarrow \Omega_f.$$

This family includes all the MRF and minimax entropy models for texture [262]. For example, a type of descriptive model for texture chooses  $\phi_j(\mathbf{I})$  as the histograms of responses from some Gabor filters.

*The second choice is the mixture family,* which can be derived by integration or summation over some hidden variables  $W = (w_1, \dots, w_K)$ ,

$$p(\mathbf{I}; \Theta) = \int p(\mathbf{I}, W; \Theta) dW = \int p(\mathbf{I}|W; \Psi) p(W; \boldsymbol{\beta}) dW.$$

The parameters of a generative model include two parts  $\Theta = (\Psi, \boldsymbol{\beta})$ . It assumes a joint probability distribution  $p(\mathbf{I}, W; \Theta)$ , and that  $W$  generates  $\mathbf{I}$  through a conditional model  $p(\mathbf{I}|W; \Psi)$  with parameters  $\Psi$ .

The hidden variables are characterized by a model  $p(W; \beta)$ .  $W$  should be *inferred* from  $\mathbf{I}$  in a probabilistic manner, and this is in contrast to the deterministic features  $\phi_k(\mathbf{I}), k = 1, \dots, K$  in descriptive models. The generative method incrementally adds hidden variables to augment the space  $\Omega_p$  and thus generates a sequence of mixture families,

$$\Omega_1^g \subset \Omega_2^g \subset \dots \subset \Omega_K^g \rightarrow \Omega_f \ni f.$$

For example, principal component analysis, wavelet image coding [25, 150], and sparse coding [136, 172] all assume a linear additive model where an image  $\mathbf{I}$  is the result of linear superposition of some window functions  $\Psi_k, k = 1, \dots, K$ , plus a Gaussian noise process  $\mathbf{n}$ .

$$\mathbf{I} = \sum_{k=1}^K a_k \Psi_k + \mathbf{n},$$

where  $a_k, k = 1, \dots, K$  are the coefficients,  $\Psi_i$  are the eigen vectors in PCA, wavelet bases in image coding, or over-complete basis for sparse coding. The hidden variables are the  $K$  coefficients of bases plus the noise, so  $W = (a_1, \dots, a_K, \mathbf{n})$ .<sup>§</sup> The coefficients are assumed to be independently and identically distributed,

$$a_k \sim p(a_k) = \text{const.} \exp^{-\lambda_o |a_k|^\rho}, \quad k = 1, \dots, K.$$

The norm  $\rho = 1$  for sparse coding [136, 172] and basis pursuit [25], and  $\rho = 2$  for principal component analysis. Thus we have a simple distribution for  $W$ ,

$$p(W; \beta) = \text{const.} \prod_{k=1}^K \exp^{-\lambda_o |a_k|^\rho} \prod_{(x,y)} \exp^{-\frac{\mathbf{n}^2(x,y)}{2\sigma_o^2}}.$$

In this example, the parameters are the  $K$  bases plus the parameters in  $p(W; \beta)$ ,  $\Theta = \{\Psi_1, \dots, \Psi_K, \lambda_o, \sigma_o\}$ . There are also occlusion models with randomly positioned discs called random collage or deadleaves models (see [130] and refs. therein).

In this model  $p(W; \beta)$  is from the exponential family. However, in the literature, hidden variables  $a_k, k = 1, \dots, K$  are assumed to be iid Gaussian or Laplacian distributed. Thus the concept of descriptive models is trivialized.

### 5.2.3 A Multi-layered Generative Model for Images

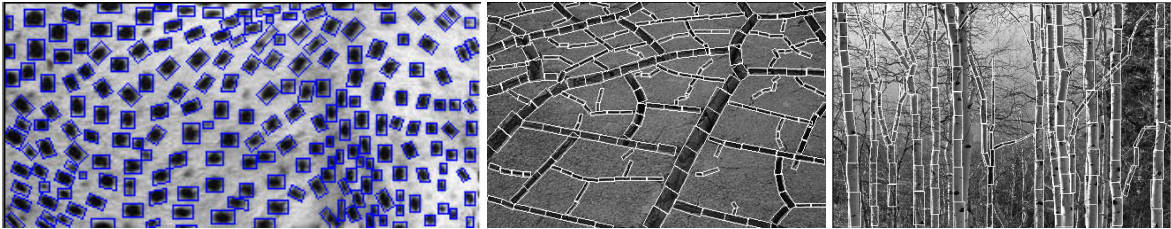


Figure 5.3: Reprinted with permission from [80]. Texture images with texton processes.

<sup>§</sup>In PCA, since the bases are orthogonal,  $a_k$  can be computed as transform, but for an over-complete basis, the  $a_k$  has to be inferred.

We focus on a multi-layer generative model for images with mainly texture content and we believe that the same learning framework can be applied to other patterns such as object shapes. An image  $\mathbf{I}$  is assumed to be generated by  $L$  layers of stochastic processes, and each layer consists of a finite number of distinct but similar elements, called “textons”. Fig 5.3 shows three typical examples of texture images, and each texton is represented by a rectangular window. A layered model is shown in Fig 5.2.

Textons at layer  $l$  are image patches transformed from a square template  $\Psi_l$ . The  $j$ -th texton in layer  $l$  is identified by six transformation variables,

$$t_{lj} = (x_{lj}, y_{lj}, \sigma_{lj}, \tau_{lj}, \theta_{lj}, A_{lj}), \quad (5.4)$$

where  $(x_{lj}, y_{lj})$  represents the texton center location,  $\sigma_{lj}$  the scale (or size),  $\tau_{lj}$  the “shear” (aspect ratio of height versus width),  $\theta_{lj}$  the orientation, and  $A_{lj}$  for photometric transforms such as lighting variability.

$t_{lj}$  defines an affine transform denoted by  $G[t_{lj}]$ , and the pixels covered by a texton  $t_{lj}$  is denoted by  $D_{lj}$ . Thus the image patch  $\mathbf{I}_{D_{lj}}$  of a texton  $t_{lj}$  is

$$\mathbf{I}_{D_{lj}} = G[t_{lj}] \odot \Psi_l, \quad \forall j, \forall l,$$

where  $\odot$  denotes the transformation operator. Texton examples of a circular template at different scales, shears, and orientations are shown in Fig. 5.4.

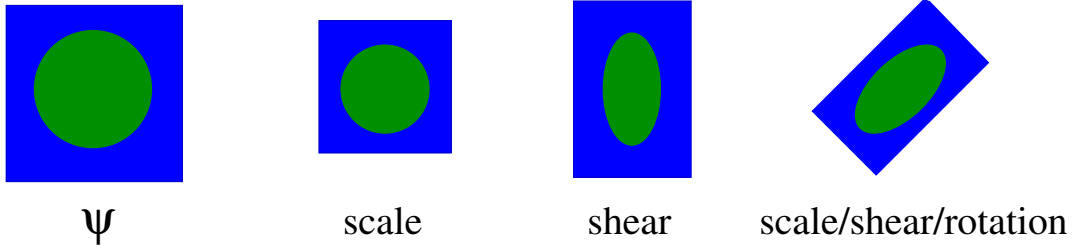


Figure 5.4: Reprinted with permission from [80]. A template  $\Psi$  (leftmost) and its three transformed copies.

We define the collection of all textons in layer  $l$  as a *texton map*,

$$\mathbf{T}_l = (n_l, \{t_{lj}, j = 1 \dots n_l\}), l = 1 \dots L,$$

where  $n_l$  is the number of textons in layer  $l$ .

In each layer, the texton map  $\mathbf{T}_l$  and the template  $\Psi_l$  generate an image  $\mathbf{I}_l = \mathbf{I}(\mathbf{T}_l; \Psi_l)$  deterministically. If several texton patches overlap at site  $(x, y)$  in  $\mathbf{I}_l$ , the pixel value is taken as average,

$$\mathbf{I}_l(x, y) = \frac{\sum_{j=1}^{n_l} \delta((x, y) \in D_{lj}) \mathbf{I}_{D_{lj}}(x, y)}{\sum_{j=1}^{n_l} \delta((x, y) \in D_{lj})},$$

where  $\delta(\bullet) = 1$  if  $\bullet$  is true, otherwise  $\delta(\bullet) = 0$ . In image  $\mathbf{I}_l$ , pixels not covered by any texton patches are transparent. The image  $\mathbf{I}$  is generated in the following way,

$$\mathbf{I}(\mathbf{T}; \Psi) = \mathbf{I}(\mathbf{T}_1; \Psi_1) \oslash \mathbf{I}(\mathbf{T}_2; \Psi_2) \oslash \dots \oslash \mathbf{I}(\mathbf{T}_L; \Psi_L), \quad \text{and} \quad \mathbf{I}^{\text{obs}} = \mathbf{I}(\mathbf{T}; \Psi) + \mathbf{n}. \quad (5.5)$$

The symbol  $\oslash$  denotes occlusion (or linear addition), i.e.  $\mathbf{I}_1 \oslash \mathbf{I}_2$  means  $\mathbf{I}_1$  occludes  $\mathbf{I}_2$ .  $\mathbf{I}(\mathbf{T}; \Psi)$  is called a *reconstructed image* and  $\mathbf{n}$  is a Gaussian noise process  $\mathbf{n}(x, y) \sim N(0, \sigma_0^2), \forall (x, y)$ . Thus pixel value at site  $(x, y)$  in the image  $\mathbf{I}$  is the same as the top layer image at that point, while uncovered pixels are only modeled by noises.

In this generative model, the hidden variables are

$$\mathbf{T} = (L, \{(\mathbf{T}_l, d_l) : l = 1, \dots, L\}, \mathbf{n}),$$

where  $d_l$  indexes the order (or relative depth) of the  $l$ -th layer.

To simplify computation, we assume that  $L = 2$  and the two stochastic layers are called “background” and “foreground” respectively. The two texton process  $\mathbf{T}_l, l = 1, 2$  are assumed to be independent of each other. We find that this assumption holds true for most of the texture patterns. Otherwise one has to make  $L$  as an unknown complexity parameter in the model.

Thus the likelihood for an observable image  $\mathbf{I}$  can be computed

$$p(\mathbf{I}; \Theta) = \int p(\mathbf{I}|\mathbf{T}; \Psi) p(\mathbf{T}; \beta) d\mathbf{T}, \quad (5.6)$$

$$= \int p(\mathbf{I}|\mathbf{T}_1, \mathbf{T}_2; \Psi) \prod_{l=1}^2 p(\mathbf{T}_l; \beta_l) d\mathbf{T}_1 d\mathbf{T}_2. \quad (5.7)$$

Let  $\Psi = (\Psi_1, \Psi_2)$  be texton templates and  $\beta = (\beta_1, \beta_2)$  the parameters for the two texton processes which we shall discuss in the next section, and  $\sigma^2$  the variance of the noise. The generative part of the model is a conditional probability  $p(\mathbf{I}|\mathbf{T}_1, \mathbf{T}_2; \Psi)$ ,

$$p(\mathbf{I}^{\text{obs}}|\mathbf{T}_1, \mathbf{T}_2; \Psi) \propto \exp \frac{-\|\mathbf{I}^{\text{obs}} - \mathbf{I}(\mathbf{T}_1, \mathbf{T}_2; \Psi)\|^2}{2\sigma^2}, \quad (5.8)$$

where  $\mathbf{I}(\mathbf{T}_1, \mathbf{T}_2; \Psi)$  is the reconstructed image from the two hidden layers without noise (see eq. (5.5)). As the generative model is very simple, the texture pattern should be captured by the spatial arrangements of textons in models  $p(\mathbf{T}_l; \beta_l), l = 1, 2$ , which are in much lower dimensional spaces and are more semantically meaningful than previous Gibbs models on pixels [262].

In the next section, we discuss the model  $p(\mathbf{T}_l; \beta_l), l = 1, 2$  for the texton processes.

## 5.2.4 A Descriptive Model of Texton Processes

As the texton processes  $\mathbf{T}_l$  are not generated by further hidden layers in the model,<sup>¶</sup> they must be characterized by descriptive models in exponential families. In this section, we first review some background on three physical ensembles, and then introduce a Gestalt ensemble for the texton process. Finally, we show some experiments for realizing the texton processes.

### Background: The physics foundation for visual modeling

There are two main differences between a texton process  $\mathbf{T}_l$  and a conventional texture defined on a lattice  $D \subset \mathbb{Z}^2$ .

- A texton process has an unknown number of elements and each element has many attributes  $t_{lj}$ , while a texture image has a fixed number of pixels and each pixel has only one variable for intensity.
- The neighborhood of a texton can change depending on their relative positions, scales, and orientations, while pixels always have fixed neighborhoods.

---

<sup>¶</sup>We may introduce additional layers of hidden variables for curve processes that render the textons. But our model stops at the texton level.

Although a texton process is more complicated than a texture image, they share a common property that they all have a large number of elements and global patterns arise from simple local interactions between elements. Thus a well-suited theory for studying these patterns is statistical physics — a subject studying macroscopic properties of a system involving a huge number of elements [23].

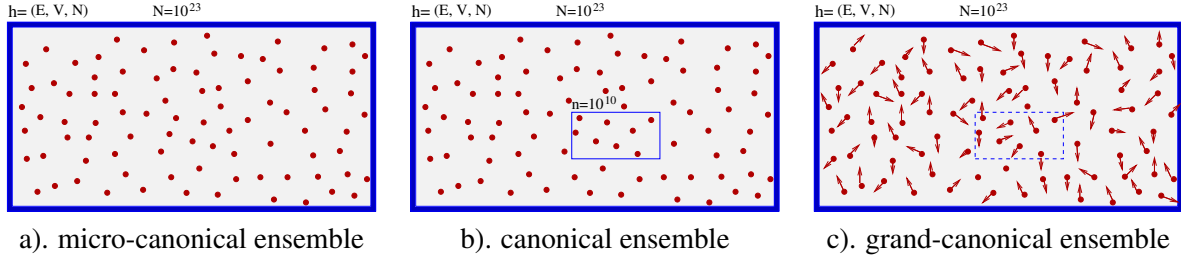


Figure 5.5: Three typical ensembles in statistical mechanics.

To understand the intuitive ideas behind various texture and texton models, we find it revealing to discuss three physical ensembles which are shown in Fig. 5.5.

1). *Micro-canonical ensemble*. Figure 5.5.a) is an insulated system of  $N$  elements. The elements could be atoms or molecules in systems such as solid ferro-magnetic material, fluid, or gas.  $N$  is nearly infinity, say  $N = 10^{23}$ . The system is decided by a configuration  $S = (\mathbf{x}^N, \mathbf{m}^N)$ , where  $\mathbf{x}^N$  describes the coordinates of the  $N$  elements and  $\mathbf{m}^N$  their momenta. The system is subject to some global constraints  $\mathbf{h}_o = (N, E, V)$ . That is, the number of elements  $N$ , the total system energy  $E$ , and total volume  $V$  is fixed. When it reaches equilibrium, this insulated system is characterized by a so-called *micro-canonical ensemble*,

$$\Omega_{mcn} = \{S : \mathbf{h}(S) = \mathbf{h}_o, f(S; \mathbf{h}_o) = 1/|\Omega_{mcn}|\}.$$

$S$  is a microscopic state or *instance*, and  $\mathbf{h}(S)$  is the macroscopic **summary** of the system. The state  $S$  is assumed to be uniformly distributed within  $\Omega_{mcn}$ , thus it is associated with a probability  $f(S; \mathbf{h}_o)$ . The system is identified by  $\mathbf{h}_o$ .

2). *Canonical ensemble*. Figure 5.5.b) illustrates a small subsystem embedded in a micro-canonical ensemble. The subsystem has  $n \ll N$  elements, fixed volume  $v \ll V$  and energy  $e$ . It can exchanges energy through the wall with the remaining elements which is called the “heat bath” or “reservoir”. At thermodynamic equilibrium, the microscopic state  $s = (\mathbf{x}^n, \mathbf{m}^n)$  for the small system is characterized by a canonical ensemble with a Gibbs model  $p(s; \beta)$ ,

$$\Omega_{cn} = \{s; p(s; \beta) = c \cdot \exp\{-\beta e(s)\}\}.$$

In our previous paper on texture modeling [233], the micro-canonical ensemble is mapped to a *Julesz ensemble* where  $S = \mathbf{I}$  is an infinite image on 2D plane  $\mathbb{Z}^2$ , and  $\mathbf{h}_o$  is a collection of Gabor filtered histograms. The canonical ensemble is mapped to a FRAME model [262] with  $s = \mathbf{I}_D$  being an image on a finite lattice  $D$ . Intuitively,  $s$  is a small patch of  $S$  viewed from a window  $D$ . The intrinsic relationship between the two ensembles is that the Gibbs model  $p(s; \beta)$  in  $\Omega_{cn}$  is derived as a conditional distribution of  $f(S; \mathbf{h}_o)$  in  $\Omega_{mcn}$ . There is a duality between  $\mathbf{h}_o$  and  $\beta$  (see [233] and refs therein).

3). *Grand-Canonical ensemble*. Figure 5.5.c) illustrates a third system where the subsystem is open and can exchange not only energy but also elements with the bath. So  $v$  is fixed, but  $n$  and  $e$  may change. This models liquid or gas materials. At equilibrium, the microscopic state  $s$  for this small system is governed by a distribution  $p(s; \beta_o, \beta)$  with  $\beta_o$  controlling the density of elements in  $s$ . Thus a grand-canonical ensemble is

$$\Omega_{gd} = \{s = (n, \mathbf{x}^n, \mathbf{m}^n); p(s; \beta_o, \beta)\}$$



The grand-canonical ensemble is a mathematical model for visual patterns with varying numbers of elements, thus lays the foundation for modeling texton processes. In the next subsection, we map the grand-canonical ensemble to a *Gestalt ensemble* in visual modeling.

### The Gestalt ensemble

Without loss of generality, we represent a spatial pattern by a set of attributed elements called textons as was discussed in section (5.2.3). To simplify notation, we consider only one texton layer on a lattice  $D$ ,

$$\mathbf{T} = (n, \{t_j = (x_j, y_j, \sigma_j, \tau_j, \theta_j, A_j), j = 1, \dots, n\}).$$

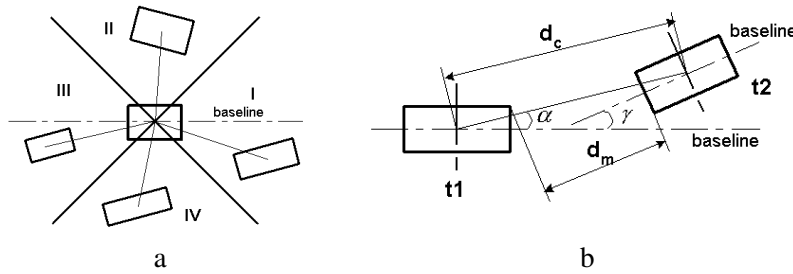


Figure 5.6: Reprinted with permission from [80]. Texton neighborhood. a). a texton has four neighbors; b). Four measurements between texton  $t_1$  and its neighbor  $t_2$ ,  $d_c$ ,  $d_m$ ,  $\alpha$ , and  $\gamma$ .

For a texton map  $\mathbf{T}$ , we define a neighborhood system  $\partial(\mathbf{T})$ .

$$\partial(\mathbf{T}) = \{\partial t : t \in \mathbf{T}, \partial t \subset \mathbf{T}\}$$

where  $\partial t$  is a set of neighboring textons for each texton  $t$ . We find the nearest neighbors are often enough. Because each texton covers a  $15 \times 15$  patch on average, a pair of adjacent textons captures image features at the scale of often more than  $30 \times 30$  pixels.

There are a few different ways of defining  $\partial(\mathbf{T})$ . One may treat each texton as a point, and compute a Voronoi diagram or Delaunay triangularization which provides graph structures for the neighborhood. For example, a Voronoi neighborhood was used in (Ahuja and Tuceryan 1989) [4] for grouping dot patterns. However, for textons, we need to consider other attributes such as orientation in defining the neighborhood. Figure 5.6.a) shows a texton  $t$ . The plane is separated into four quadrants relative to the two axes of the rectangle. In each quadrant, the nearest texton is considered as the neighbor texton. Unlike the Markov random field on image lattice, the texton neighborhood is no longer translation invariant.

The above neighborhood is defined deterministically. In more general settings,  $\partial(\mathbf{T})$  shall be represented by a set of hidden variables that can be inferred from  $\mathbf{T}$ . Thus a texton may have a varying number of neighbors referenced by some indexing (or address) variables. These address variables could be decided probabilistically depending on the relative positions, orientations, and scales or intensities. This leads to the so-called *mixed Markov random field* and is beyond the scope of this section. Mumford and Fridman discussed such cases in another context (see [56]).

For a texton  $t_1$  and its neighbor  $t_2 \in \partial t$ , we measure five features shown in Figure 5.6.b, which capture various Gestalt properties:

1.  $d_c$ : Distance between two centers, which measures *proximity*.

2.  $d_m$ : Gap between two textons, which measures connectedness and continuation.
3.  $\alpha$ : Angle of a neighbor relative to the main axis of the reference texton. This is mostly useful in quadrants I and III.  $\alpha/d_c$  measures the curvature of possible curves formed by the textons, or *co-linearity and co-circularity* in the Gestalt language.
4.  $\gamma$ : Relative orientations between the two textons. This is mostly useful for neighbors in quadrants II and IV and measures *parallelism*.
5.  $r$ : Size ratio which denotes the similarity of texton sizes.  $r$  is the width of  $t_2$  divided by the width of  $t_1$  for neighbors into quadrants I and III and  $r$  is the length of  $t_2$  divided by the length of  $t_1$  for neighbors in quadrants II and IV.

Thus a total of  $4 \times 5 = 20$  pairwise features are computed for each texton plus two features of each texton itself: The orientation  $\theta_j$  and a two-dimensional feature the scaling and shearing  $(\sigma_j, \tau_j)$ . Following the notation of descriptive models in section (5.2.2), we denote these features by

$$\phi^{(k)}(t|\partial t), \quad \text{for } k = 1, \dots, 22.$$

We compute 21 one dimensional marginal histograms and one two-dimensional histogram for  $(\sigma_j, \tau_j)$ , averaged over all textons.

$$H^{(k)}(z) = \sum_{j=1}^n \delta(z - \phi^{(k)}(t_j|\partial t_j)), \quad \forall k.$$

We denote these histograms by

$$H(\mathbf{T}) = (H^{(1)}, \dots, H^{(22)}), \quad \text{and} \quad \mathbf{h}(\mathbf{T}) = \frac{1}{n} H(\mathbf{T}).$$

The vector length of  $\mathbf{h}(\mathbf{T})$  is the total number of bins in all histograms. One may choose other features and high order statistics as well. In the vision literature, (Steven, 1978) was perhaps the earliest attempt for characterizing spatial patterns using histogram of attributes (See [153] for some examples).

The distribution of  $\mathbf{T}$  is characterized by an statistical ensemble in correspondence to the grand-canonical ensemble in Fig. 5.5.c. We call it a *Gestalt ensemble* on a finite lattice  $D$  as it is the general representation for various Gestalt patterns,

$$\text{A Gestalt ensemble} = \Omega_{gst} = \{\mathbf{T} : p(\mathbf{T}; \beta_o, \beta)\}. \quad (5.9)$$

The Gestalt ensemble is governed by a Gibbs distribution,

$$p(\mathbf{T}; \beta_o, \beta) = \frac{1}{Z} \exp\{-\beta_o n - \langle \beta, H(\mathbf{T}) \rangle\}. \quad (5.10)$$

$Z$  is a partition function.  $\beta_o$  is a parameter controlling texton density. We can rewrite the vector valued potential functions  $\beta$  as energy functions  $\beta^{(k)}()$ , then we have

$$p(\mathbf{T}; \beta_o, \beta) = \frac{1}{Z} \exp\{-\beta_o n - \sum_{j=1}^n \sum_{k=1}^{K=22} \beta^{(k)}(\phi^{(k)}(t_j|\partial t_j))\},$$

This model provides a rigorous way for integrating multiple feature statistics into one probability model, and generalizes existing point processes [26].

The probability  $p(\mathbf{T}; \beta_o, \beta)$  is derived from the Julesz ensemble (or micro-canonical ensemble). We first define a close system with  $N \gg n$  elements on a lattice  $D$ , and we assume the density of textons is fixed

$$\lim_{N \rightarrow \infty} \frac{N}{|D|} = \rho, \quad \text{as } N \rightarrow \infty, \text{ and } D \rightarrow \mathbb{Z}^2.$$

Thus we obtain a Julesz ensemble on  $\mathbb{Z}^2$  [233],

$$A \text{ Julesz ensemble} = \Omega_{jlz} = \{\mathbf{T}_\infty : \mathbf{h}(\mathbf{T}_\infty) = \mathbf{h}_o, N \rightarrow \infty, f(\mathbf{T}_\infty; \mathbf{h}_o)\},$$

where  $\mathbf{h}_o = (\rho, \mathbf{h})$  is the macroscopic summary of the system state  $\mathbf{T}_\infty$ . On any finite image, a texton process should be a conditional density of  $f(\mathbf{T}_\infty; \mathbf{h}_o)$ . There is a one-to-one correspondence between  $\mathbf{h}_o = (\rho, \mathbf{h})$  and the parameters  $(\beta_o, \beta)$ .

We can learn the parameters  $(\beta_o, \beta)$  and select effective features  $\phi^{(k)}$  by the descriptive method — the minimax entropy learning paradigm [262]. In the following subsection, we discuss some computational issues and experiments for learning  $p(\mathbf{T}; \beta_o, \beta)$  and simulating the Gestalt ensembles.

### 5.2.5 An Integrated Learning Framework

After discussing the descriptive models for the hidden texton layers, we now return to the integrated framework presented in section (5.2.3).

#### The Integrated Learning Paradigm

The generative model for an observed image  $\mathbf{I}^{\text{obs}}$  is rewritten from equation (5.7),

$$p(\mathbf{I}^{\text{obs}}; \Theta) = \int p(\mathbf{I}^{\text{obs}} | \mathbf{T}_1, \mathbf{T}_2; \Psi) \prod_{l=1}^2 p(\mathbf{T}_l; \beta_l) d\mathbf{T}_1 d\mathbf{T}_2. \quad (5.11)$$

We follow the ML-estimate in equation (11.52),

$$\Theta^* = \arg \max_{\Theta \in \Omega_K^g} \log p(\mathbf{I}^{\text{obs}}; \Theta).$$

The parameters  $\Theta$  include the texton templates  $\Psi_l$ , the Lagrange multipliers  $\beta_l$ ,  $l = 1, 2$  for two Gestalt ensembles, and the variance of the Gaussian noise,  $\sigma^2$

$$\Theta = (\Psi, \beta, \sigma), \quad \Psi = (\Psi_1, \Psi_2), \quad \text{and} \quad \beta = (\beta_{10}, \beta_1, \beta_{20}, \beta_2).$$

To maximize the log-likelihood, we take the derivative with respect to  $\Theta$ , and set it to zero. Let  $\mathbf{T} = (\mathbf{T}_1, \mathbf{T}_2)$ ,

$$\begin{aligned} & \frac{\partial \log p(\mathbf{I}^{\text{obs}}; \Theta)}{\partial \Theta} \\ &= \int \frac{\partial \log p(\mathbf{I}^{\text{obs}}, \mathbf{T}; \Theta)}{\partial \Theta} p(\mathbf{T} | \mathbf{I}^{\text{obs}}; \Theta) d\mathbf{T} \\ &= \int \left[ \frac{\partial \log p(\mathbf{I}^{\text{obs}} | \mathbf{T}; \Psi)}{\partial \Psi} + \sum_{l=1}^2 \frac{\partial \log p(\mathbf{T}_l; \beta_l)}{\partial \beta_l} \right] p(\mathbf{T} | \mathbf{I}^{\text{obs}}; \Theta) d\mathbf{T} \end{aligned}$$

$$= E_{p(\mathbf{T}|\mathbf{I}^{\text{obs}}, \Theta)} \left[ \frac{\partial \log p(\mathbf{I}^{\text{obs}}|\mathbf{T}; \Psi)}{\partial \Psi} + \sum_{l=1}^2 \frac{\partial \log p(\mathbf{T}_l; \beta_l)}{\partial \beta_l} \right] = 0.$$

In the literature, there are two well-known methods for solving the above equation. One is the EM algorithm [36], and the other is data augmentation [209] in the Bayesian context. We propose to use a stochastic gradient algorithm [79] which is more effective for our problem.

#### *A Stochastic Gradient Algorithm*

– **Step 0.** Initialize the hidden texton maps  $\mathbf{T}$  and the templates  $\Psi$  using a simplified likelihood as discussed in the next section. Set  $\beta = 0$ .

Repeat steps I and II below iteratively (like EM-algorithm).

– **Step I.** With the current  $\Theta = (\Psi, \beta, \sigma)$ , obtain a sample of texton maps from the posterior probability

$$\mathbf{T}_m^{\text{syn}} \sim p(\mathbf{T}|\mathbf{I}^{\text{obs}}; \Theta) \propto p(\mathbf{I}^{\text{obs}}|\mathbf{T}_1, \mathbf{T}_2; \Psi) p(\mathbf{T}_1; \beta_{1o}, \beta_1) p(\mathbf{T}_2; \beta_{2o}, \beta_2), \quad m = 1, \dots, M. \quad (5.12)$$

This is *Bayesian inference*. The sampling process is realized by a Monte Carlo Markov chain which simulates a random walk with two types of dynamics.

- I.a). A *diffusion dynamics* realized by a Gibbs sampler — sampling (relaxing) the transform group for each texton. For example, move textons, update their scales and rotate them etc.
- I.b). A *jump-dynamics* — adding or removing a texton (death/birth) by reversible jumps [75].

— **Step II.** We treat  $\mathbf{T}_m^{\text{syn}}, m = 1, \dots, M$  as “observations”, and estimate the integration in eq. (5.12) by importance sampling. Thus we have

$$\frac{\partial \log p(\mathbf{I}^{\text{obs}}|\mathbf{T}; \Psi)}{\partial \Psi} + \sum_{l=1}^2 \frac{\partial \log p(\mathbf{T}_l; \beta_l)}{\partial \beta_l} = 0$$

We learn  $\Theta = (\Psi, \beta, \sigma)$  of the texton templates and Gibbs models respectively by gradient ascent:

- II.a). Update the texton templates  $\Psi$  by maximizing  $\sum_{m=1}^M \log p(\mathbf{I}^{\text{obs}}|\mathbf{T}_m^{\text{syn}}; \Psi)$ ; this is a fitting process. In our experiment, the texton templates  $\Psi_1, \Psi_2$  are represented by  $15 \times 15$  windows and thus there are  $2 \times 225$  unknowns.<sup>†</sup>
- II.b). Update  $\beta_{1o}, \beta_l, l = 1, 2$  by maximizing  $\sum_{m=1}^M \log p(\mathbf{T}_m^{\text{syn}}; \beta_{1o}, \beta_l)$ . This is exactly the maximum entropy learning process in the descriptive method except that the texton processes are given by step I.
- II.c). Update  $\sigma$  for the noise process.

In step I, we choose to sample  $M = 1$  example each time. If the learning rate in steps II.a) and II.b) is slow enough, the expectation is estimated by importance sampling through samples  $\mathbf{T}^{\text{syn}}$  over time. It has been proved in statistics [79] that such an algorithm converges to the optimal  $\Theta$  if the step size in step II satisfies some mild conditions.

The following are some useful observations.

1. Descriptive models and learning is part of the integrated learning framework, in terms of both representation and computing (Step II.b)).

---

<sup>†</sup>Each point in the window can be transparent, and thus the shape of the texton can change during the learning process.

2. Bayesian vision inference is a sub-task (step I) of the integrated learning process. A vision system, machine or biological, evolves by learning generative models  $p(\mathbf{I}; \Theta)$  and makes an inference about the world  $\mathbf{T}$  (or  $W$  in more general generative models) using the current imperfect knowledge  $\Theta$  — the Bayesian view of vision. What is missing in this learning paradigm is a “discovery process” that introduces new hidden variables.

### Mathematical definitions of visual patterns in the integrated learning paradigm

Any visual learning paradigm must answer the question of *conceptualization*: How do we define, mathematically, a visual *concept* or a pattern? For example, a human face, a wood grain texture, and so on. In this section, we show how the descriptive and generative methods conceptualize a visual pattern.

The *concept* of a visual pattern is an *abstraction* (or *summary*) for an assembly of configurations (or instances)  $s$  that are not distinguished by human perception for a certain vision *purpose*. Because of the stochastic nature of the visual signal, instances in this assembly are governed by a frequency  $f(s)$ , and an instance can then be considered a random sample from the probability distribution  $f(s)$ . Thus a concept is said to be equal to an *ensemble*,

$$\text{a visual concept } c = \Omega_c = \{s : f(s)\}.$$

By a descriptive method, the ensemble is defined through statistical constraints (see equation.(5.3)),

$$\text{a descriptive concept } c = \Omega(\mathbf{h}_c) = \{s : f(s; \mathbf{h}_c), E_f[\mathbf{h}(s)] = \mathbf{h}_c\}.$$

For example,  $s$  could be a human face represented by a list of key points, or  $s = \mathbf{I}$  could be a texture image.  $\mathbf{h}(s)$  is the statistics extracted from  $s$  which are sufficient for a certain vision purpose.  $\mathbf{h}_c$  is vector value that identifies this concept.  $\mathbf{h}_c$  corresponds to a Gibbs model  $p(s; \beta)$  with parameter  $\beta$ . It is accepted that two concepts may have overlapping ensembles.

When the signal is homogeneous, such as texture or texton maps, the expectation  $E_f[\mathbf{h}(s)]$  can be computed from a single instance through spatial average over a large enough lattice  $D$ , thus we can define a concept as an equivalence class — called the Julesz ensemble in (Zhu et al. 2000) [205].

$$\text{a descriptive concept } c = \Omega(\mathbf{h}_c) = \{s : \mathbf{h}(s) = \mathbf{h}_c \text{ } D \rightarrow \mathbb{Z}^2, f(s; \mathbf{h}_c)\},$$

For example, for defining a texture,  $\mathbf{h}_c$  is the sufficient and necessary statistics extracted in texture perception. The ensembles on large lattice  $D \rightarrow \mathbb{Z}^2$  are disjoint and deterministic as  $f(s; \mathbf{h}_c)$  is a uniform distribution.

Though the pure descriptive concept and Julesz ensemble are technically sound, they are only a coarse or first stage approximation to human perception in texture discrimination. Texture studies in psychology [109, 114] suggest that human vision is sensitive to the perception of some basic elements called textons, perhaps for some vision purpose. It was also argued and demonstrated by Malik et al. [134, 147, 148] that the detection of individual textons plays an important role in texture discrimination, segmentation and grouping. However, it was unclear what textons really are.

We argue that the mathematical definition of textons must be guided by a generative model. So the integrated learning paradigm extends the Julesz ensemble definition to a generative concept with  $W$  being the hidden variables,

$$\text{a generative concept } c = \{s : f(s; \Theta) = \int f(s|W; \Psi) f(W; \mathbf{h}_w) dW\},$$

where  $W$  are instances of a descriptive ensemble,

$$\{W : \mathbf{h}(W) = \mathbf{h}_w, D \rightarrow Z^2, f(W; \mathbf{h}_w)\}.$$

Thus a concept is identified by the parameters  $\Theta = (\Psi, \mathbf{h}_w)$  (or  $\Theta = (\Psi, \beta)$ ), because of the duality between  $\mathbf{h}_w$  and  $\beta$ ). In this definition,  $\Psi$  is a mathematically sound definition of texton.

### 5.2.6 Effective Inference by Simplified Likelihood

In this section, we address some computational issues in the integrated learning paradigm, and propose a method for initializing the stochastic gradient algorithm (in step 0).

#### Initialization by likelihood simplification and clustering

The stochastic algorithm presented in section (5.2.5)) needs a long “burn-in” period if it starts from an arbitrary condition. To accelerate the computation, we use a simplified likelihood in step 0 of the stochastic gradient algorithm. Thus given an input image  $\mathbf{I}^{\text{obs}}$ , our objective is to compute some good initial texton templates  $\Psi_1, \Psi_2$  and hidden texton maps  $\mathbf{T}_1, \mathbf{T}_2$ , before the iterative process in steps I and II.

A close analysis reveals that the computational complexity is largely due to the complex coupling between the textons in both the generative model  $p(\mathbf{I}|\mathbf{T}_1, \mathbf{T}_2; \Psi)$  and the descriptive models  $p(\mathbf{T}_1; \beta_{1o}, \beta_1)$  and  $p(\mathbf{T}_2; \beta_{2o}, \beta_2)$ . Thus we simplify both models by decoupling the textons.

Firstly, we decouple the textons in  $p(\mathbf{T}_1; \beta_{1o}, \beta_1)$  and  $p(\mathbf{T}_2; \beta_{2o}, \beta_2)$ . We fix the total number of textons  $n_1 + n_2$  to an excessive number, thus we do not need to simulate the death-birth process. We set  $\beta_1$  and  $\beta_2$  to 0, therefore  $p(\mathbf{T}_l; \beta_{lo}, \beta_l)$  becomes a uniform distribution and the texton elements are decoupled from spatial interactions.

Secondly, we decouple the textons in  $p(\mathbf{I}_{\text{obs}}|\mathbf{T}_1, \mathbf{T}_2; \Psi)$ . Instead of using the image generating model in eq. (5.5) which implicitly imposes couplings between texton elements through eq. (5.8), we adopt a constraint-based model

$$p(\mathbf{I}^{\text{obs}}|\mathbf{T}, \Psi) \propto \exp\left\{-\sum_{l=1}^2 \sum_{j=1}^{n_l} \|\mathbf{I}_{D_{lj}}^{\text{obs}} - G[T_{lj}] \odot \Psi_l\|^2 / 2\sigma^2\right\}, \quad (5.13)$$

where  $\mathbf{I}_{D_{lj}}^{\text{obs}}$  is the image patch of the domain  $D_{lj}$  in the observed image. For pixels in  $\mathbf{I}^{\text{obs}}$  not covered by any textons, a uniform distribution is assumed to introduce a penalty.

We run the stochastic gradient algorithm on the decoupled log-likelihood, which reduces to a conventional clustering problem. We start with two random texton maps and the algorithm iterates the following two steps.

I). Given  $\Psi_1$  and  $\Psi_2$ , the algorithm runs a Gibbs sampler to change each texton  $t_{lj}$  respectively, by moving, rotating, scaling the rectangle, and changing the cluster into which each texton falls according to the simplified model of eq. (5.13). Thus the texton windows intend to cover the entire observed image, and at the same time try to form tight clusters around  $\Psi$ .

II). Given  $\mathbf{T}_1$  and  $\mathbf{T}_2$ , the algorithm updates the texton  $\Psi_1$  and  $\Psi_2$  by averaging

$$\Psi_l = \frac{1}{n_l} \sum_{j=1}^{n_l} G^{-1}[T_{lj}] \odot \mathbf{I}_{D_{lj}}^{\text{obs}}, \quad l = 1, 2,$$

where  $G^{-1}[T_{lj}]$  is the inverse transformation. The layer order  $d_1$  and  $d_2$  are not needed for the simplified model.

This initialization algorithm for computing  $(\mathbf{T}_1, \mathbf{T}_2, \Psi_1, \Psi_2)$  resembles the transformed component analysis [55]. It is also inspired by a clustering algorithm by (Leung and Malik, 1999) [135], which did not engage hidden variables, and thus compute a variety of textons  $\Psi$  at different scale and orientations. We also experimented with representing the texton template  $\Psi$  by a set of Gabor bases instead of a  $15 \times 15$  window. However, the results were not as encouraging as in this generative model.

### Experiment I: Texton clustering

In the following we demonstrate experiments for initialization and clustering as section (5.2.6) stated.

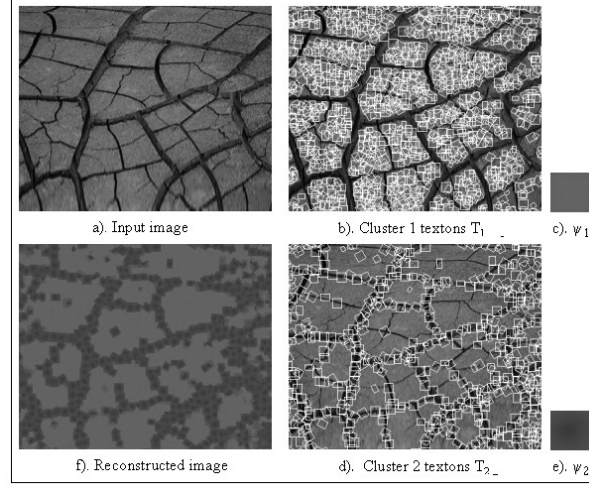


Figure 5.7: Reprinted with permission from [80]. Result of the initial clustering algorithm.

Fig. 5.7 shows an experiment on the initialization algorithm for a crack pattern. 1055 textons are used with the template size of  $15 \times 15$ . The number of textons is as twice as necessary to cover the whole image. In optimizing the likelihood in eq. (5.13), an annealing scheme is utilized with the temperature decreasing from 4 to 0.5. The sampling process converges to a result shown in Fig. 5.7.

Fig. 5.7.a) is the input image; Fig. 5.7.b) and Fig. 5.7.d) are the texton maps  $\mathbf{T}_1$  and  $\mathbf{T}_2$  respectively. Fig. 5.7.c) and Fig. 5.7.e) are the cluster centers  $\Psi_1$  and  $\Psi_2$ , shown by rectangles respectively. Fig. 5.7.f) is the reconstructed image. The results demonstrate that the clustering method provides a rough but reasonable starting solution for generative modeling.

### Experiment II: Integrated learning and synthesis

Next, we show experimental results obtained by the integrated learning paradigm. For an input image, we first do a clustering step as section (8.2.3) showed. Then we run the stochastic gradient algorithm on the full models to refine the clustering results.

Fig. 5.8 shows the result for the crack image obtained by the stochastic gradient algorithm, which took about 80 iterations of the two steps (Step I and Step II), following the initial solution (Step 0) shown in Fig. 5.7. Fig. 5.8.b) and Fig. 5.8.d) are the background and foreground texton maps  $\mathbf{T}_1$  and  $\mathbf{T}_2$  respectively. Fig. 5.8.c) and Fig. 5.8.e) are the learned textons  $\Psi_1, \Psi_2$  respectively. Fig. 5.8.f) is the reconstructed image from learned Texton maps and templates. Compared to the results in Fig. 5.7, the results in Fig. 5.8 have more precise texton maps and accurate texton templates due to an accurate generative model. The foreground texton  $\Psi_2$  is a bar, and one pixel at corner of the left-top is transparent.

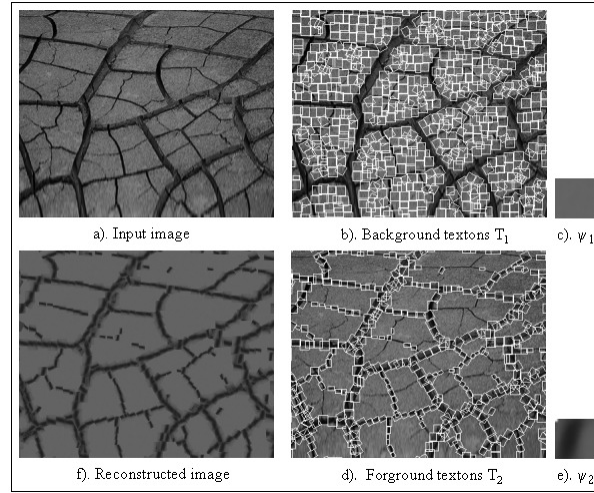


Figure 5.8: Reprinted with permission from [80]. Generative model learning result for the crack image. a) input image, b) and d) are background and foreground textons discovered by the generative model, c) and e) are the templates for the generative model, f) is the reconstructed image from the generative model.

The integrated learning results for a cheetah skin image are shown in Fig. 5.9. It can be seen that in the foreground template, the surround pixels are learned as being transparent and the blob is exactly computed as the texton. Fig. 5.10 are the results for a brick image. No point in the template is transparent for the gap lines between bricks.

Figure 5.11 shows the learning of another short crack patterns. Fig. 5.12 displays a pine corn pattern. The seeds and the black intervals are separated cleanly, and the reconstructed image keeps most of the pine structures. However the pine corn seeds are learnt as the background textons and the gaps between pine corns are treated as foreground textons.

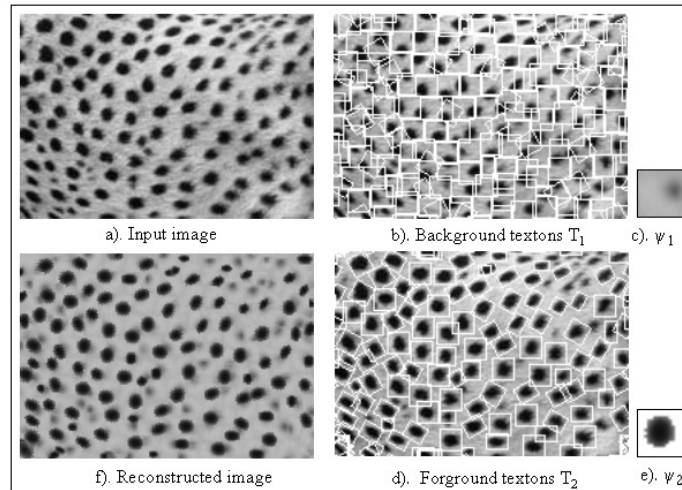


Figure 5.9: Reprinted with permission from [80]. Generative model learning result for a cheetah skin image. The notations are the same as in Fig. 5.8.

After the parameters  $\Psi$  and  $\beta$  of a generative model are discovered for a type of texture images, new



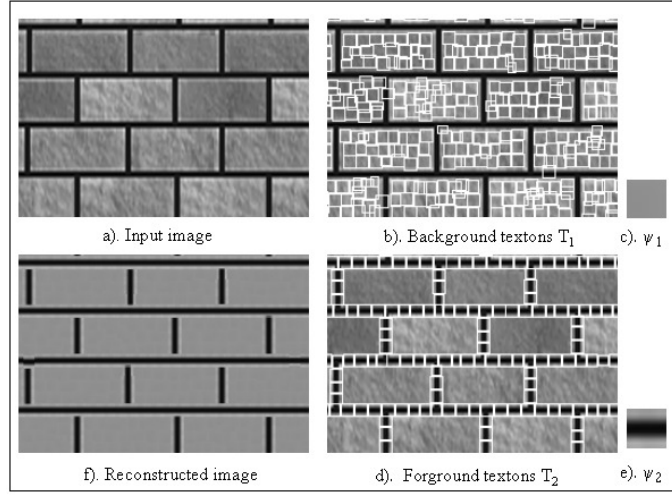


Figure 5.10: Reprinted with permission from [80]. Generative model learning result for a brick image. The notations are the same as in Fig. 5.8.

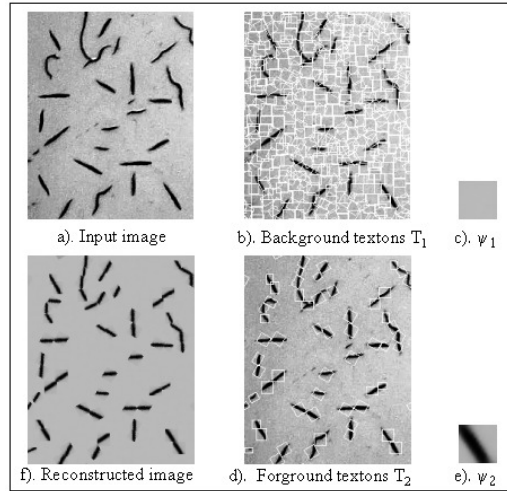


Figure 5.11: Reprinted with permission from [80]. Generative model learning result for a crack image. The notations are the same as in Fig. 5.8.

random samples could be drawn from the generative model. This proceeds in three steps: First, texton maps are sampled from the Gibbs models  $p(\mathbf{T}_1; \beta_1)$  and  $p(\mathbf{T}_2; \beta_2)$  respectively. Second, background and foreground images are synthesized from the texton maps and texton templates. Third, the final image is generated by combining these two images according the occlusion model.

We shown synthesis experiments on three patterns.

1. Fig 5.13 and Fig. 5.14 are two synthesis examples of the two layered model synthesis for the cheetah skin pattern. The templates used here are the learned results in Fig 5.9.
2. Figure 5.16 shows texture synthesis for the crack pattern computed in Figure 5.11.
3. Figure 5.16 displays texture synthesis for the brick pattern in Figure 5.10.

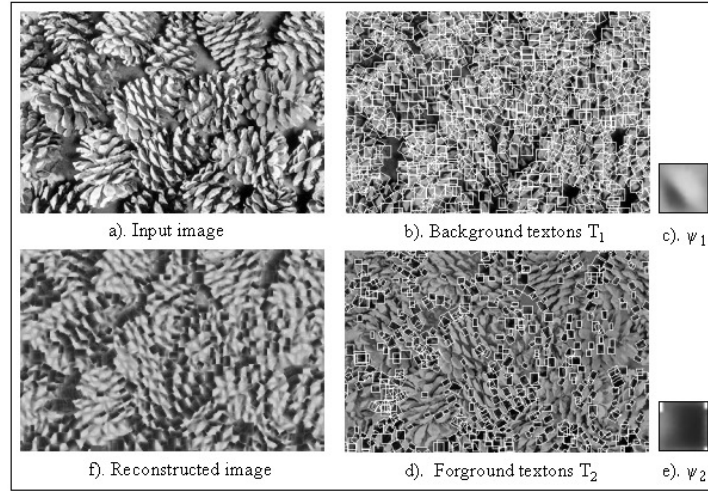


Figure 5.12: Reprinted with permission from [80]. Generative model learning result for a pine corn image. The notations are the same as in Fig. 5.8.

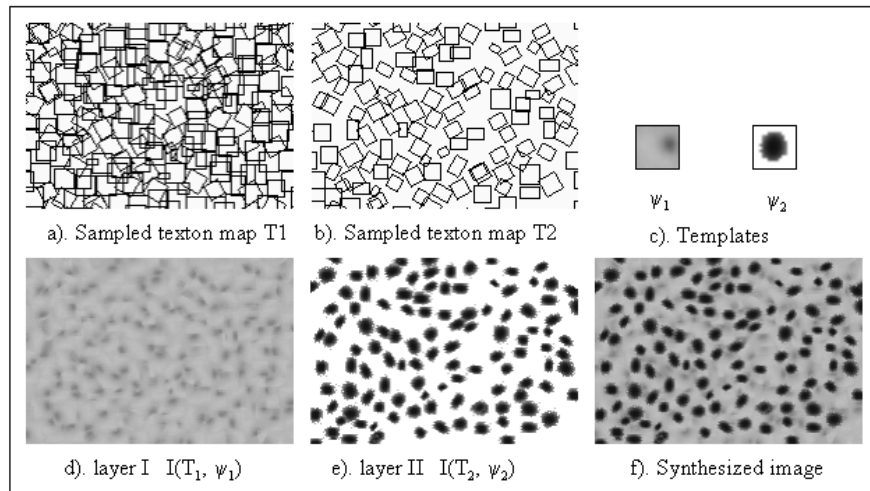


Figure 5.13: Reprinted with permission from [80]. An example of a randomly synthesized cheetah skin image. a) and b) are the background and foreground texton maps sampled from  $p(\mathbf{T}_l; \beta_l)$ ; d) and e) are synthesized background and foreground images from the texton map and templates in c); f) is the final random synthesized image from the generative model.

Note that, in these texture synthesis experiments, the Markov chain operates with meaningful textons instead of pixels.

### 5.2.7 Discussion

We present a visual learning paradigm that integrates and extends descriptive and generative models, and which also provides a sound framework for visual conceptualization and for defining textons. The hierarchical model for textures has advantages over the previous pure descriptive method with Markov random fields on pixel intensities.

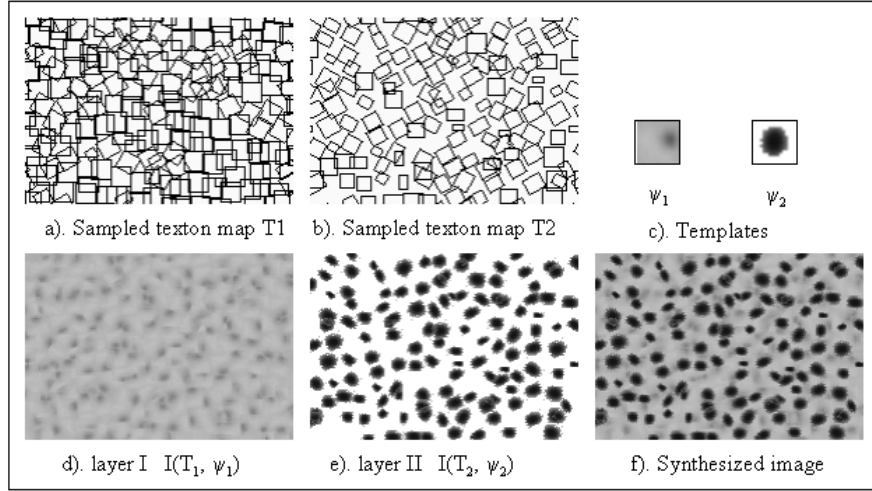


Figure 5.14: Reprinted with permission from [80]. Second example of a randomly synthesized cheetah skin image. Notation is the same as in Fig. 5.13.

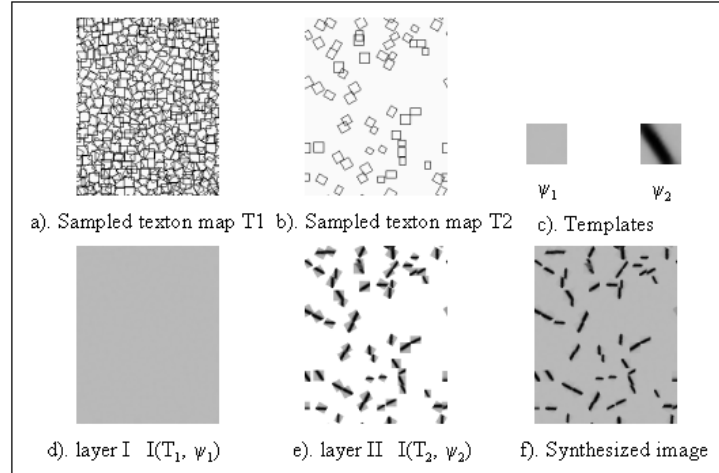


Figure 5.15: Reprinted with permission from [80]. An example of a randomly synthesized crack image. Notations are the same as in Fig. 5.13.

Firstly, from the representational perspective, the neighborhood in the texton map is much smaller than the pixel neighborhood in a FRAME model [262]. The generative method captures more semantically meaningful elements on the texton maps.

Secondly, from the computational perspective, the Markov chain operating the texton maps can move textons according to affine transforms and can add or delete a texton by death/birth dynamics, thus it is much more effective than the Markov chain used in traditional Markov random fields which flips the intensity of one pixel at a time.

We show that the integration of descriptive and generative methods is a natural path for visual learning. We argue that a vision system should evolve by progressively replacing descriptive models with generative models, which realizes a transition from *empirical and statistical models* to *physical and semantical models*. The work presented here provides a step towards this goal.

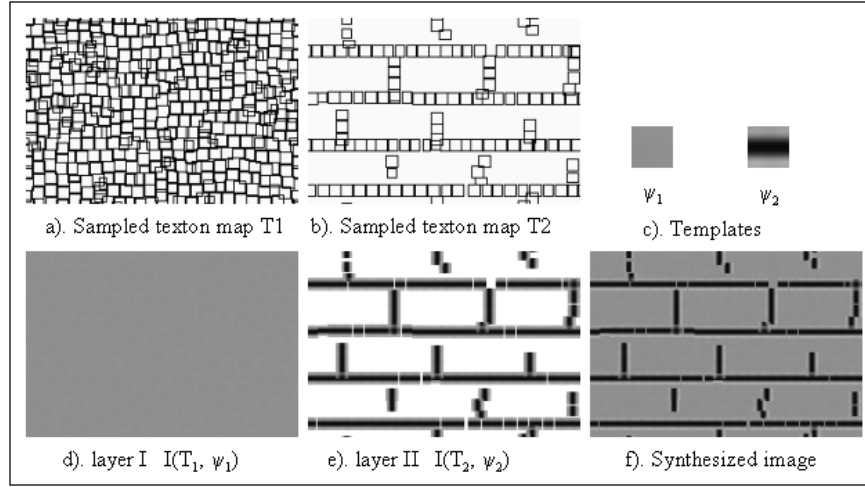


Figure 5.16: Reprinted with permission from [80]. An example of a randomly synthesized brick image. Notation is the same as in Fig. 5.13.

In future research, we shall develop effective algorithms for sampling the Gestalt ensembles, and extend the generative model to curve and graph processes. These models will be employed for general purpose vision tasks, such as segmentation and grouping.

## 6

# Primal Sketch: Integrating Textures and Textons

In his monumental book [153], Marr inherited Julesz's texton [114] notion and proposed the concept of image primitives as basic perceptual tokens, such as edges, bars, junctions, and terminators. Inspired by the Nyquist sampling theorem in signal processing, Marr went a step further and asked for a token representation which he named "primal sketch" as a perceptually lossless conversion from the raw image. He tried to reconstruct the image with zero-crossings unsuccessfully and his effort was mostly limited by the lack of proper models of texture.

## 6.1 Marr's Conjecture on Primal Sketch

In the early stage of vision perception, an image may be divided into two components – the structural part with noticeable elements called "textons" by Julesz or "image primitives" by Marr, and the textural part without distinguishable elements in preattentive vision. The structural part is often composed of objects, such as tree twigs and trunks, at a near distance whose positions and shapes can be clearly perceived. In contrast, the textural part is composed of objects at a far distance whose structures become indistinguishable and thus yield various texture impressions.

The modeling of texture and structure has been a long-standing puzzle in the study of early vision. In the 1960s, Julesz first proposed a texture theory and conjectured that a texture is a set of images sharing some common statistics on some features related to human perception. Later he switched to a texton theory and identified bars, edges, terminators as textons – the atomic elements in early vision. Marr summarized Julesz's theories along with experimental results and proposed a primal sketch model in his book as a "symbolic" or "token" representation in terms of image primitives. Marr argued that this symbolic representation should be parsimonious and sufficient to reconstruct the original image without much perceivable distortion.

In fact, Andrew Glennerster [40] shows that the primal sketch representation provides a stable coordinate frame under rapid eye rotation, which is biologically plausible. Once the certain task is defined, the visual system extracts 'raw' visual information from the primal sketch representation to higher-level information.

Despite many inspiring observations, Marr's description provided neither an explicit mathematical formulation nor a rigorous definition of the primal sketch model.

Since the 1980s, the studies of early image modeling followed two distinct paths which represent two prevailing mathematical theories for generative image modeling respectively. In fact, the two theories are two distinct ways for learning image manifolds residing in different entropy regimes respectively. In the

following, we should briefly review the two theories. Interested readers are recommended to review the last chapter for a more complete account of generative and descriptive models.

The first theory is a two layer generative model originated from computational harmonic analysis which represents images by a linear superposition of image bases selected from a dictionary – often over-complete like various wavelets, image pyramids, and sparse coding. Each image base is supposed to represent some image features with hidden variables describing their locations, orientations, scales, and intensity contrast. The image is reconstructed with minimum error on the pixel intensity.

The second theory is the Markov random fields (MRF) originated from statistical mechanics. It represents a visual pattern by pooling the responses of a bank of filters over all locations into some statistical summary like the histograms which are supposed to represent our texture impressions. On large image lattices, a Julesz ensemble is defined as a perceptual equivalence class where all images in the set share identical statistics. The statistics or texture impression are the *macroscopic* properties and the differences between *microscopic* states (i.e. image instances in the Julesz ensemble) are ignored. In other words, all images in this equivalence class are perceptually the same, replacing one with the other does not cause perceivable distortion, although the two images have large a difference in pixel by pixel comparison. The image patches within local windows are shown to follow some MRF models called FRAME.

## 6.2 The Two Layer Model

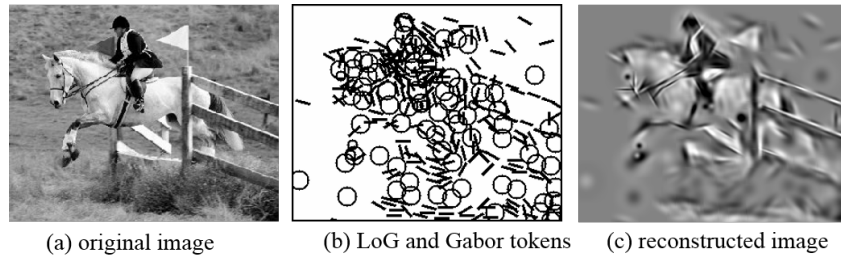


Figure 6.1: Reprinted with permission from [81]. A sparse coding example computed by matching pursuit. (b) is a symbolic representation where each base  $B_k$  is represented by a bar at the same location, with the same elongation and orientation. The isotropic LOG bases are represented by a circle.

According to the model, the image is generated as a mosaic as follows: the image lattice is divided into two disjoint parts: a structured domain, or a sketchable part and a textured domain, or a non-sketchable part,

$$D = D_{\text{sk}} \cup D_{\text{nsk}}, D_{\text{sk}} \cap D_{\text{nsk}} = \phi. \quad (6.1)$$

The image intensities on the structure domain are represented by a set of coding functions for edges and ridges. The image intensities on the texture domain are characterized by Markov random fields that interpolate the structure domain of the image. See Figures 6.3 and 6.4 for two examples of primal sketch model.

The motivation is that in the space of image patches, there are simple geometric primitives such as edges, but there are also stochastic texture patterns, as illustrated by Figure ???. We need both to describe an image.

### 6.2.1 Structure Domain

The structured domain  $D_{\text{sk}}$  is further divided into a number of disjoint patches with each patch being fitted by an image primitive

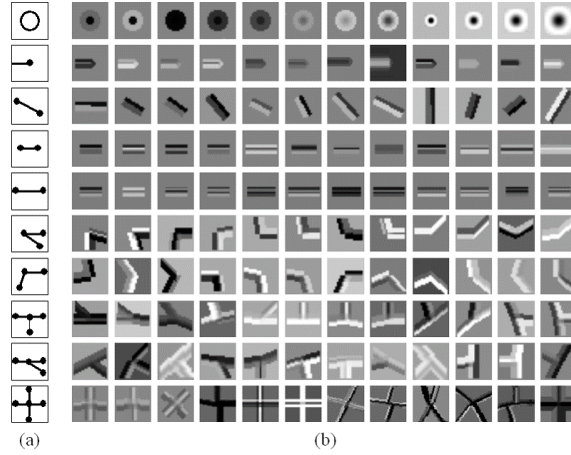


Figure 6.2: Reprinted with permission from [82]. A collection of local structure elements employed by the model. There are eight types of elements: blobs, end points, edges, ridges, multi-ridges, corners, junctions and crosses. (a) The symbolic representation. (b) The photometric representation.

$$\Lambda_{\text{sk}} = \cup_{i=1}^K \Lambda_{\text{sk},i}; \Lambda_{\text{sk},i} \cap \Lambda_{\text{sk},j} = \phi, i \neq j. \quad (6.2)$$

Some examples of the image primitive are shown in Figure 6.2. These primitives are aligned through their landmarks to form a sketch graph  $S_{\text{sk}}$ . Specifically, we index the selected image primitives by  $i = 1, \dots, n$ , and denote image patch for primitive  $i$  as  $B(x, y|\theta_i)$ . Here

$$\theta_i = (\theta_{\text{topological},i}, \theta_{\text{geometric},i}, \theta_{\text{photometric},i}), \quad (6.3)$$

where  $\theta_{\text{topological},i}$  is the type (degree of arms) of the primitive (blob, terminator, corner, junctions, etc.),  $\theta_{\text{geometric},i}$  collects the locations of the landmarks of the primitive, and  $\theta_{\text{photometric},i}$  collects the intensity profiles of the arms of the primitive. The sketch graph is a layer of hidden representation, which has to be inferred from the image

$$S_{\text{sk}} = ((D_{\text{sk},i}, B(x, y|\theta_i), a_i), i = 1, \dots, n), \quad (6.4)$$

where  $S_{\text{sk}}$  decides the structure domain of the image, and  $a_i$  is the address variable pointing to the neighbors of the vertex  $S_{\text{sk},i} = (D_{\text{sk},i}, B(x, y|\theta_i))$ .

The model for the structure domain of the image is

$$\mathbf{I}(x, y) = \sum_{i=1}^n B(x, y|\theta_i) + \epsilon(x, y), \quad (x, y) \in \Lambda_{\text{sk}}, \quad i = 1, \dots, n, \quad (6.5)$$

where  $\epsilon(x, y)$  denotes the noise term. Since the set of pixels coded by  $B(x, y | \theta_i)$  do not overlap each other,  $B(x, y|\theta_i)$  is similar to coding vectors in vector quantization.

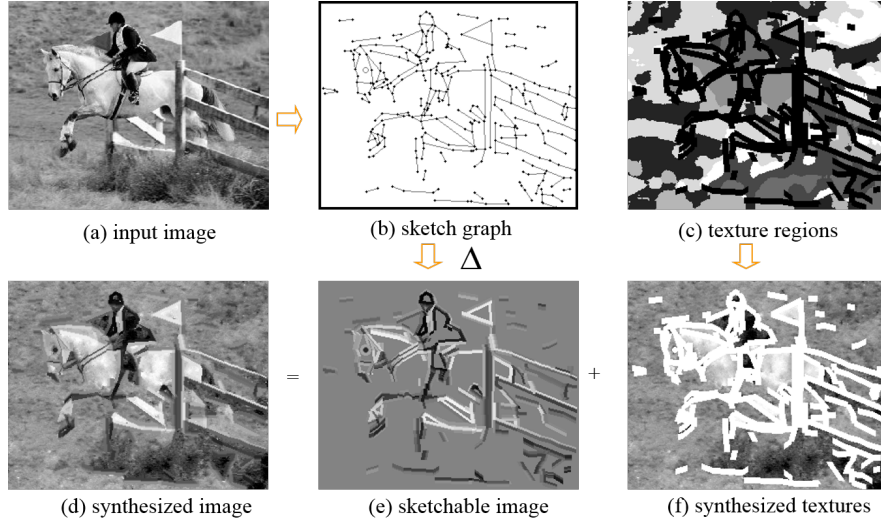


Figure 6.3: Reprinted with permission from [82]. Primal sketch model. (a) Observed image. (b) “Sketchable” part is described by a geometric sketch graph. (c) The texture regions of the image. (d) Fill in the “non-sketchable” part by matching feature statistics. (e) The sketchable part of the image. (f) The non-sketchable part of the image.

## 6.2.2 The Dictionary of Image Primitives

An edge segment is modeled by a 2D function that is constant along the edge, and has a profile across the edge. Specifically,

$$B(x, y \mid \theta) = f(-(x - u) \sin \alpha + (y - v) \cos \alpha), \quad (6.6)$$

where

$$\begin{aligned} -l &< (x - u) \cos \alpha + (y - v) \sin \alpha \leq l, \\ -w &\leq -(x - u) \sin \alpha + (y - v) \cos \alpha \leq w. \end{aligned}$$

That is, the function  $B(x, y \mid \theta)$  is supported on a rectangle centered at  $(u, v)$ , with length  $2l + 1$ , width  $2w + 1$ , and orientation  $\alpha$ .

For the profile function  $f(x)$ , let  $f_0(x) = -1/2$  for  $x < 0$  and  $f_0(x) = 1/2$  for  $x \geq 0$ , and let  $g_s(\cdot)$  be a Gaussian function of standard deviation  $s$ . Then  $f(x) = a + bf_0(x) * g_s(x)$ . This is the model proposed by Elder and Zucker. The convolution with Gaussian kernel is used to model the blurred transition of intensity values across the edge, caused by the three dimensional shape of the underlying physical structure, as well as the resolution and focus of the camera. As proposed by Elder and Zucker, the parameter  $s$  can be determined by the distance between the two extrema of the second derivative  $f''(x)$ .

Thus in the coding function  $B(x, y \mid \theta)$  for an edge segment,  $\theta = (t, u, v, \alpha, l, w, s, a, b)$ , namely, type (which is edge in this case), center, orientation, length, width, sharpness, average intensity, intensity jump.  $\theta$  captures geometric and photometric aspects of an edge explicitly, and the coding function is non-linear in  $\theta$ .



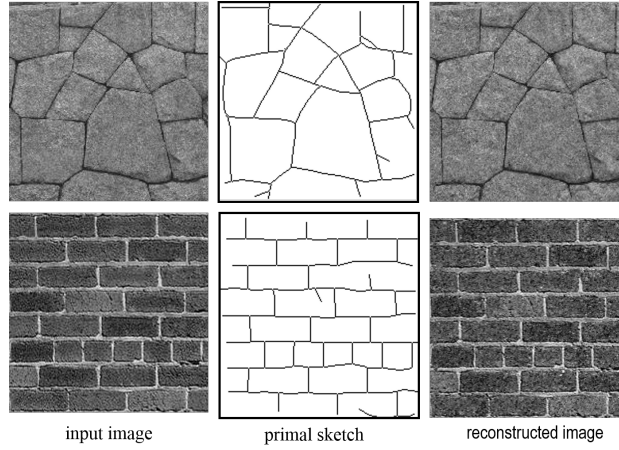


Figure 6.4: Reprinted with permission from [82]. Examples of primal sketch model. (a) Observed image. (b) Sketch graph. (c) Synthesized image from the fitted model.

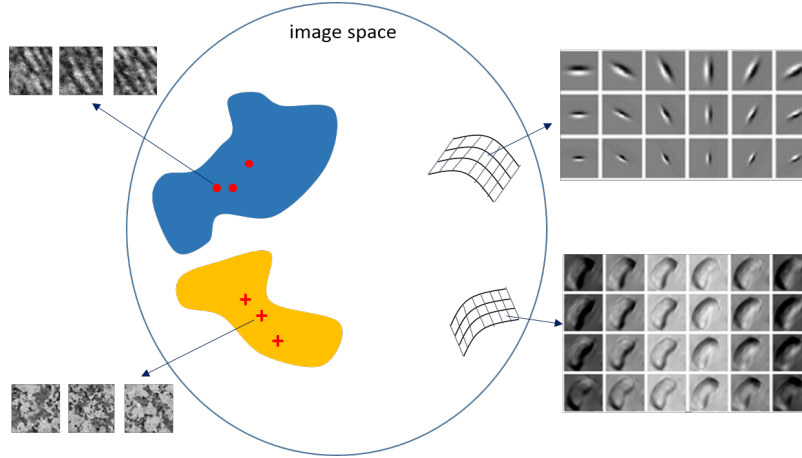


Figure 6.5: In the space of image patches, there are simple shape primitives, and there are also stochastic texture patterns.

A ridge segment has the same functional form, where the profile  $f(x)$  is a composition of two edge profiles. The profile of a multi-ridge is a composition of two or more ridge profiles. A blob function is modeled by rotating an edge profile, more specifically,  $B(x, y | \theta) = f(\sqrt{(x-u)^2 + (y-v)^2} - r)$ , where  $(x-u)^2 + (y-v)^2 \leq R^2$ , and again  $f(x) = a + bf_0(x) * g_s(x)$  being a step edge convolved with a Gaussian kernel. This function is supported on a disk area centered at  $(u, v)$  with radius  $R$ . The transition of intensity occurs at the circle of radius  $r < R$ .

The corners and junctions are important structures in images. They are modeled as compositions of edge or ridge functions. When a number of such coding functions join to form a corner or a junction, the image intensities of the small number of overlapping pixels are modeled as averages of these coding functions. The endpoint of a ridge is modeled by a half blob.

See Figure (6.2) for a sample of local structure elements, which are the coding functions and their combinations. There are eight types of elements: blobs, endpoints, edges, ridges, multi-ridges, corners, junctions, and crosses. Figure (6.2.a) shows the symbolic representations of these elements. Figure (6.2.b)

displays the image patches of these elements.

Let  $S_{\text{sk}}$  be the sketch graph formed by these coding functions. The graph has a set of nodes or vertices  $V = \cup_{d=0}^4 V_d$ , where  $V_d$  is the set of nodes with degree  $d$ , i.e., the nodes with  $d$  arms. For instance, a blob node has degree 0, an endpoint has degree 1, a corner has degree 2, a T-junction has degree 3, and across has degree 4. We do not allow nodes with more than 4 arms.  $S_{\text{sk}}$  is regularized by a simple spatial prior model:

$$p(S_{\text{sk}}) \propto \exp\left\{-\sum_{d=0}^4 \lambda_d |V_d|\right\}, \quad (6.7)$$

where  $|V_d|$  is the number of nodes with  $d$  arms. The prior probability or the energy term  $\gamma_{\text{sk}}(S_{\text{sk}}) = \sum_{d=0}^4 \lambda_d |V_d|$  penalizes free end points by setting  $\lambda_{\text{sk}}$  at a large value.

### 6.2.3 Texture Domain

The texture domain  $\Lambda_{\text{nsk}}$  is segmented into  $m$  regions of homogenous texture patterns,

$$\Lambda_{\text{nsk}} = \cup_{j=1}^m \Lambda_{\text{nsk},j}; \Lambda_{\text{nsk},i} \cap \Lambda_{\text{nsk},j} = \phi, i \neq j. \quad (6.8)$$

Within each region  $j$ , we pool the marginal histograms of the responses from the  $K$  filters,  $h_j = (h_{j,k}, k = 1, \dots, K)$ , where

$$h_{j,k,z} = \frac{1}{|\Lambda_{\text{nsk},j}|} \sum_{(x,y) \in \Lambda_{\text{nsk},j}} \delta(z; F_k * \mathbf{I}(x, y)), \quad (6.9)$$

where  $z$  indexes the histogram bins, and  $\delta(z; x) = 1$  if  $x$  belongs to bin  $z$ , and  $\delta(z; x) = 0$  otherwise. This yields a Markov random field model, also known as descriptive model, for each texture region:

$$p(\mathbf{I}_{\Lambda_{\text{nsk},j}}) \propto \exp\left\{-\sum_{(x,y) \in \Lambda_{\text{nsk},j}} \sum_{k=1}^K \phi_{j,k}(F_k * \mathbf{I}(x, y))\right\}. \quad (6.10)$$

These Markov random fields have the structure domain as boundary conditions, because when we apply filters  $F_k$  on the pixels in  $\Lambda_{\text{nsk}}$ , these filters may also cover some pixels in  $\Lambda_{\text{sk}}$ . These Markov random fields in-paint the texture domain  $\Lambda_{\text{nsk}}$  while interpolating the structure domain  $\Lambda_{\text{sk}}$ , and the in-painting is guided by the marginal histograms of linear filters within each region.

Let  $S_{\text{nsk}} = ((\Lambda_{\text{nsk},j}, h_{j,k} \leftrightarrow \phi_{j,k}), j = 1, \dots, m, k = 1, \dots, K)$  denotes the segmentation of the texture domain.  $S_{\text{nsk}}$  follows a prior model  $p(S_{\text{nsk}}) \propto \exp\{-\gamma_{\text{nsk}}(S_{\text{nsk}})\}$ , for instance,  $\gamma_{\text{nsk}}(S_{\text{nsk}}) = \rho m$  to penalize the number of regions.

### 6.2.4 Integrated Model

Formally, we can integrate the structure model (6.5) and the texture model (6.10) into a probability distribution. Our inspiration for such integration comes from the model of Mumford and Shah [162]. In their method, the prior model for the noiseless image can be written as

$$p(\mathbf{I}, S) = \frac{1}{Z} \exp\left\{-\sum_{(x,y) \in \Lambda/S} \lambda |\nabla \mathbf{I}(x, y)|^2 - \gamma |S|\right\}, \quad (6.11)$$

where  $S$  is a set of pixels of discontinuity that correspond to the boundaries of objects, and  $|S|$  is the number of pixels in  $S$ . In model (6.11),  $S$  is the structure domain of the image, and the remaining part is the texture domain.

Our model can be viewed as an extension of the Mumford-Shah model. Let  $S = (S_{\text{sk}}, S_{\text{nsk}})$ , we have

$$p(\mathbf{I}, S) = \frac{1}{Z} \exp \left\{ - \sum_{i=1}^n \sum_{(x,y) \in \Lambda_{\text{sk},i}} \frac{1}{2\sigma^2} (\mathbf{I}(x,y) - B_i(x,y | \theta_i))^2 - \gamma_{\text{sk}}(S_{\text{sk}}) \right. \\ \left. - \sum_{j=1}^m \sum_{(x,y) \in \Lambda_{\text{nsk},j}} \sum_{k=1}^K \phi_{j,k}(F_k * \mathbf{I}(x,y)) - \gamma_{\text{nsk}}(S_{\text{nsk}}) \right\}. \quad (6.12)$$

Compared to Mumford-Shah model, model (6.12) is more sophisticated in both the structure part and texture part.

### 6.2.5 The Sketch Pursuit Algorithm

To learn the integrated model, the traditional maximum likelihood algorithm requires the MCMC method for global inference. Instead, a greedy algorithm, sketch pursuit algorithm is proposed. It consists of the following phases:

1. *Phase 0*: an edge and ridge detector based on linear filters are run to give an initialization for the sketch graph.
2. *Phase 1*: a greedy algorithm is used to determine the sketch graph but without using the spatial prior model.
3. *Phase 2*: a greedy algorithm based on a set of graph operators is used to edit the sketch graph to achieve good spatial organization as required by the spatial prior model.
4. *Phase 3*: the remaining portion of the image is segmented into homogeneous texture regions by clustering the local marginal histograms of filter responses. The inference algorithm yields two outputs:
  - (a) a sketch graph for the image, with edge and ridge segments, as well as corners and junctions.
  - (b) a parameterized representation of the image which allows the image to be re-synthesized and to be encoded efficiently.

## 6.3 Hybrid Image Templates

The primal sketch model is developed for generic scenes. In this section, we study a more specific framework for learning representations for objects from images, the hybrid image templates HIT.

First, just as the primal sketch, the appearances of objects have two types of descriptors: local sketch and texture gradient. For instance, the image in Figure 6.6 consists of both the patches of geometric primitives on the object boundary and the patches of textures on the object surface. In addition to sketch features and texture features, we also add flatness features and color features, so that the templates give complete descriptions of the images. Fig. 6.7 shows some examples of such templates, with local sketch (edge or bar), texture gradients (with orientation field), flatness regions (smooth surface and lighting), and colors.

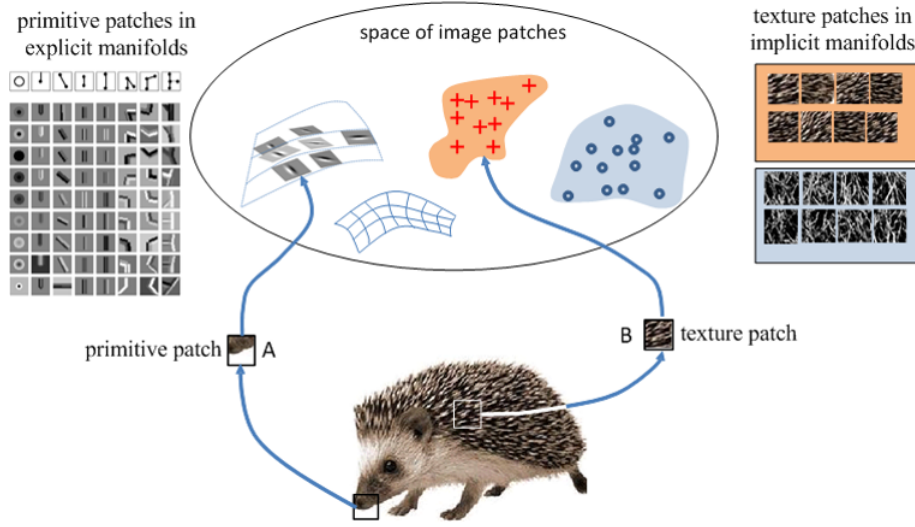


Figure 6.6: Reprinted with permission from [199]. An image of an object consists of patches of simple primitives and patches of textures.

The modeling and learning strategy of the hybrid template is similar to the active basis. For each type of feature, we pool a background histogram  $q$  from natural images and then estimate the distribution  $p$  by exponential tilting. We select the features sequentially as in active basis. After each feature is selected, it will inhibit nearby features of the same type.

Naturally, there are large variations in the representations of different classes, for example, teapots may have a common shape outline, but do not have a common texture or color. The hedgehog in Figure 6.6 has a distinct texture and shape, but its color is often less distinguishable from its background. The essence of our learning framework is to automatically select, in a principled way, informative patches from a large pool and compose them into a template with a normalized probability model.

### 6.3.1 Representation

Let  $\Lambda$  be the image lattice for the object template which is typically  $150 \times 150$  pixels. This template will undergo a similarity transform to align with object instances in images. The lattice is decomposed into a set of  $K$  patches  $\{\Lambda_k, k = 1, 2, \dots, K\}$  selected from a large pool in the learning process through feature pursuit. These patches belong to four bands: sketch, texture/gradient field, flatness, and color respectively, and do not form a partition of the lattice  $\Lambda$  for two reasons:

- Certain pixels on  $\Lambda$  are left unexplained due to inconsistent image appearances at these positions.
- Two selected patches from different bands may overlap each other in position. For example, a sketch patch and a color patch can occupy the same region, but we make sure the sketch feature descriptor and color descriptor extracted from them would represent largely uncorrelated information.

The hybrid image template consists of the following components,

$$\text{HIT} = (\{\Lambda_k, \ell_k, \{B_k \text{ or } \mathbf{h}_k\}, \delta_k : k = 1, 2, \dots, K\}, \Theta) \quad (6.13)$$

where

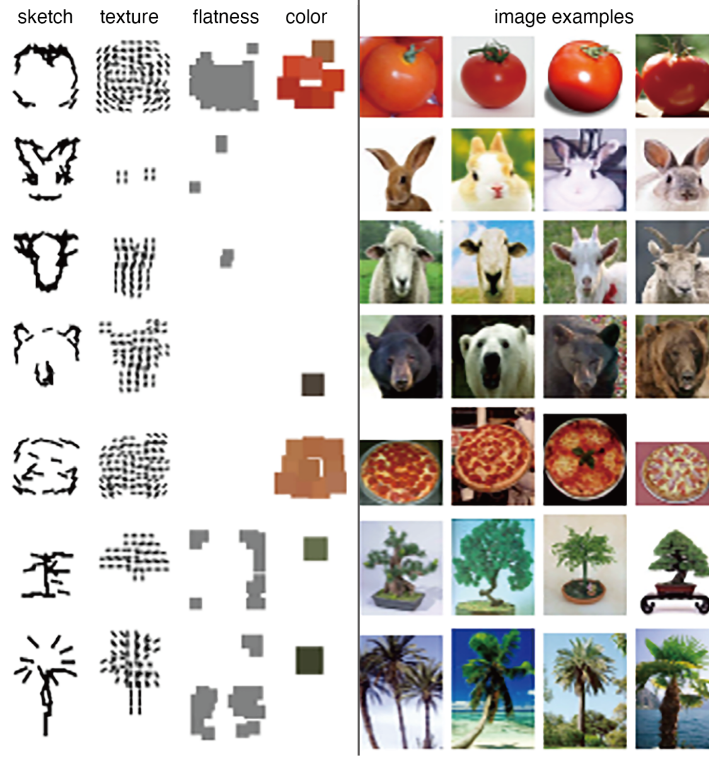


Figure 6.7: Reprinted with permission from [199]. The hybrid templates consist of sketch, texture, flatness, and color features.

1.  $\Lambda_k \subset \Lambda$  is the  $k$ -th patch lattice described above.
2.  $\ell_k \in \{'\text{skt}', '\text{txt}', '\text{flt}', '\text{clr}'\}$  is the type of the patch.
3.  $B_k$  or  $\mathbf{h}_k$  is the feature prototype for the  $k$ -th patch. If  $\ell_k = '\text{skt}'$ , then the patch is described by a basis function  $B_k$  for the image primitive, otherwise it is described by a histogram  $\mathbf{h}_k$  for texture gradients, flatness or color respectively.
4.  $\delta_k = (\delta_k x, \delta_k y, \delta_k \theta)$ : the latent variables for the local variabilities of the  $k$ -th patch, *i.e.* the local translations and rotations of selected patches.
5.  $\Theta = \{\lambda_k, z_k : k = 1, 2, \dots, K\}$  are the parameters of the probabilistic model  $p$  (to be discussed in the subsection).  $\lambda_k, z_k$  are the linear coefficient and normalizing constant for the  $k$ -th patch.

### 6.3.2 Prototypes, $\epsilon$ - balls, and Saturation Function

Let  $\mathbf{I}_{\Lambda_k}$  be the image defined on the patch  $\Lambda_k \subset \Lambda$ . For  $\ell_k = '\text{skt}'$ , the prototype  $B_k$  defines a subspace through an explicit function for  $\mathbf{I}_{\Lambda_k}$  (a sparse coding model),

$$\Omega(B_k) = \{\mathbf{I}_{\Lambda_k} : \mathbf{I}_{\Lambda_k} = c_k B_k + \epsilon\}. \quad (6.14)$$

For  $\ell_k \in \{'\text{txt}', '\text{flt}', '\text{clr}'\}$ , the prototype defines a subspace through an implicit function for  $\mathbf{I}_{\Lambda_k}$  which constrains the histogram (a Markov random field model),

$$\Omega(\mathbf{h}_k) = \{\mathbf{I}_{\Lambda_k} : H(\mathbf{I}_{\Lambda_k}) = \mathbf{h}_k + \epsilon\}. \quad (6.15)$$

$H(\mathbf{I}_{\Lambda_k})$  extracts the histogram (texture gradient, flatness, or color) from  $\mathbf{I}_{\Lambda_k}$ . In  $\Omega(B_k)$ , the distance is measured in the image space,

$$\rho^{\text{ex}}(\mathbf{I}_{\Lambda_k}) = \|\mathbf{I}_{\Lambda_k} - cB_k\|^2 \quad (6.16)$$

while in  $\Omega(h_k)$ , the distance is measured in the projected histogram space with norm.

$$\rho^{\text{im}}(\mathbf{I}_{\Lambda_k}) = \|H(\mathbf{I}_{\Lambda_k}) - \mathbf{h}_k\|^2 \quad (6.17)$$

Intuitively, we may view  $\Omega(B_k)$  and  $\Omega(\mathbf{h}_k)$  as  $\epsilon$ -balls centered at the prototypes  $B_k$  and  $\mathbf{h}_k$ , respectively, with different metrics. Each  $\epsilon$ -ball is a set of image patches which are perceptually equivalent. Thus the image space of HIT is the product space of these heterogeneous subspaces:  $\Omega(\text{HIT}) = \prod_{k=1}^K \Omega_k$ , on which a probability model is concentrated. Due to statistical fluctuations in small patches, these  $\epsilon$ -balls have soft boundaries. Thus we use a sigmoid function to indicate whether a patch  $\mathbf{I}_{\Lambda_k}$  belongs to a ball  $\Omega(B_k)$  or  $\Omega(\mathbf{h}_k)$ .

$$r(\mathbf{I}_{\Lambda_k}) = S(\rho(\mathbf{I}_{\Lambda_k})), \quad (6.18)$$

where  $\rho$  can be either  $\rho^{\text{ex}}$  or  $\rho^{\text{im}}$ .  $S(x)$  is a saturation function with maximum at  $x = 0$ :

$$S(x) = \tau \left( \frac{2}{1 + e^{-2(\eta-x)/\tau}} - 1 \right), \quad (6.19)$$

with shape parameters  $\tau$  and  $\eta$ . We set  $\tau = 6$  and  $\eta$  is locally adaptive:  $\eta = \|\mathbf{I}_{\Lambda_k}\|^2$  where  $\mathbf{I}_{\Lambda_k}$  denotes the local image patch. We call  $r(\mathbf{I}_{\Lambda_k})$  the *response* of the feature (prototype  $B_k$  or  $\mathbf{h}_k$ ) on patch  $\mathbf{I}_{\Lambda_k}$ .

### 6.3.3 Projecting Image Patches to 1D Responses

Though the image patches are from heterogeneous subspaces of varying dimensions with different metrics, we project them into the one-dimensional feature response  $r(\mathbf{I}_{\Lambda_k})$ , on which we can calculate the statistics (expectation) of  $r(\mathbf{I}_{\Lambda_k})$  over the training set regardless of the types of patches. This way it is easy to integrate them into a probabilistic model.

In the following, we discuss the details of computing the responses for the four different image subspaces.

Given an input color image  $\mathbf{I}$  on lattice  $\Lambda$ , we first transform it into an HSV-space with HS being the chromatic information and V the gray level image. We apply a common set of filters  $\Delta$  to the gray level image. The dictionary  $\Delta$  includes Gabor filters (sine and cosine) at 3 scales and 16 orientations. The Gabor filter of the canonical scale and orientation is of the form:  $F(x, y) \propto \exp\{-(x/\sigma_1)^2 - (y/\sigma_2)^2\}e^{ix}$  with  $\sigma_1 = 5$ ,  $\sigma_2 = 10$ .

1. *Calculating responses on primitives.* When a patch  $\mathbf{I}_{\Lambda_k}$  contains a prominent primitive, such as an edge or bar, it is dominated by a filter which inhibits all the other filters. Thus the whole patch is represented by a single filter, which is called a Basis function  $B_k \in \Delta$ . The response is calculated as the local maximum over the activity  $\delta_k$ ,

$$r^{\text{skt}}(\mathbf{I}_{\Lambda_k}) = \max_{\delta x, \delta y, \delta \theta} S(\|\mathbf{I} - cB_{x+\delta x, y+\delta y, o+\delta o}\|^2). \quad (6.20)$$

The local maximum pooling is proposed by [197] as a possible function of complex cells in V1.

2 *Calculating responses on texture.* In contrast to the primitives, a texture patch usually contains many small elements, such as the patch on the hedgehog body in Figure 6.6. As a result, many filters have medium responses on the image patch. Thus we pool a histogram of these filters collectively over the local patch to form a histogram descriptor  $H(\mathbf{I})$ .

The texture response is calculated by

$$r^{\text{txt}}(\mathbf{I}_{\Lambda_k}) = S(\|H(\mathbf{I}_{\Lambda_k}) - \mathbf{h}\|^2), \quad (6.21)$$

where  $\mathbf{h}$  is a pre-computed histogram prototype (one may consider it as a cluster center of similar texture patches). More specifically,  $\mathbf{h}$  is obtained by averaging the histograms at the same position of roughly aligned positive example images. For texture, we are only interested in the medium to strong strengths along certain directions. So we replace the indicator function, which is often used in histogram binning, by a continuous function  $a(x) = \frac{12}{1+e^{-x/3}} - 6$ . The histogram is then weighted into one bin for each filter,

$$H_o(\mathbf{I}_{\Lambda_k}) = \frac{1}{|\Lambda_k|} \sum_{(x,y) \in \Lambda_k} a(|F_o * \mathbf{I}_{\Lambda_k}|^2). \quad (6.22)$$

Thus, we obtain the oriented histogram for all filters as a  $|\mathcal{O}|$ -vector,

$$H(\mathbf{I}_{\Lambda_k}) = (H_1, \dots, H_{|\mathcal{O}|}). \quad (6.23)$$

It measures the strengths in all orientations.

3. *Calculating responses on flat patch.* By flat patch we mean image area that are void of structures, especially edges. Thus filters have near-zero responses. They are helpful for suppressing false alarms in cluttered areas. As a texture-less measure, we choose a few small filters  $\Delta^{\text{flt}} = \{\nabla_x, \nabla_y, LoG\}$  and further compress the texture histogram into a single scalar,

$$H(\mathbf{I}_{\Lambda_k}) = \sum_{F \in \Delta^{\text{flt}}} \sum_{(x,y) \in \Lambda_k} b(|F_o * \mathbf{I}_{\Lambda_k}|^2). \quad (6.24)$$

$b()$  is a function that measures the featureless responses. It takes the form of a sigmoid function like  $S()$  but with different shape parameters. In Figure 6.8 we plot the four functions  $a()$ ,  $b()$ ,  $\mathbf{1}()$  and  $S()$  for comparison. Then the flatness response is defined as,

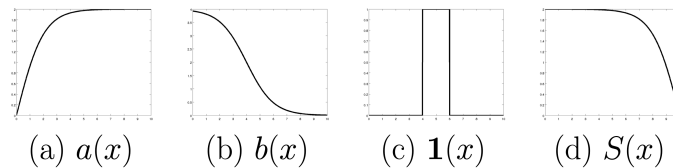


Figure 6.8: Reprinted with permission from [199]. Plotting the four functions.

$$r^{\text{flt}}(\mathbf{I}_{\Lambda_k}) = S(H(\mathbf{I}_{\Lambda_k}) - h). \quad (6.25)$$

In the above  $h = 0$  is a scalar for flatness prototype.

4. *Calculating responses on color.* The chromatic descriptors are informative for certain object categories. Similar to orientation histogram, we calculate a histogram  $H^{\text{clr}}(\mathbf{I}_{\Lambda_k})$  on the color space (we use the 2D HS-space in the HSV format). Then the color patch response is defined as the saturated distance between the color histogram of the observed image and the prototype histogram  $\mathbf{h}$ ,

$$r^{\text{clr}}(\mathbf{I}_{\Lambda_k}) = S(\|H^{\text{clr}}(\mathbf{I}_{\Lambda_k}) - \mathbf{h}\|^2). \quad (6.26)$$

In summary, a HIT template consists of  $K$  prototypes  $\{B_k \text{ or } \mathbf{h}_k, k = 1, \dots, K\}$  for sketch, texture/gradient, flatness, and color patches respectively which define  $K$ -subspaces (or  $\epsilon$ -balls)  $\Omega(B_K)$  or  $\Omega(\mathbf{h}_k)$  of varying dimensions. These  $\epsilon$ -balls quantize the image space with different metrics. An input image  $\mathbf{I}$  on lattice  $\Lambda$  is then projected to the HIT and is represented by a vector of responses:

$$\mathbf{I} \rightarrow (r_1, r_2, \dots, r_K)$$

where  $r_k$  is a soft measure for whether the image patch  $\mathbf{I}_{\Lambda_k}$  belongs to the subspace defined by the corresponding prototype. In the next section we will define a probability model on image  $\mathbf{I}$  based on these responses.

### 6.3.4 Template Pursuit by Information Projection

We present an algorithm for learning the hybrid image templates automatically from a set of image examples. It pursues the image patches, calculates their prototypes, and derive a probability model sequentially until the information gain is within the statistical fluctuation – a model complexity criterion similar to AIC.

Let  $f(\mathbf{I})$  be the underlying probability distribution for an image category, and our objective is to learn a series of models that approach  $f$  from an initial or reference model  $q$ ,

$$q = p_0 \rightarrow p_1 \rightarrow p_2 \rightarrow \dots \rightarrow p_K \approx f. \quad (6.27)$$

These models sequentially match the observed marginal statistics collected from the samples of  $f$ . With more marginal statistics matched between the model  $p$  and  $f$ ,  $p$  will approach  $f$  in terms of reducing the Kullback-Leibler divergence  $\text{KL}(f||p)$  monotonically.

The main input to the learning algorithm is a set of positive examples

$$D^+ = \{\mathbf{I}_1, \dots, \mathbf{I}_n\} \sim f,$$

where  $f$  is the underlying target image distribution and  $\sim$  means sampled from. For simplicity, we may assume these images contain roughly aligned objects that can be explained by a common HIT template. When this alignment assumption is not satisfied, we can adopt an EM-like iterative procedure with the unknown object localization as missing data. We are also given a set of negative examples

$$D^- = \{\mathbf{J}_1, \dots, \mathbf{J}_N\} \sim \text{reference distribution } q.$$

The negative examples are only used for pooling marginal histograms of one-dimensional feature responses in a pre-computation step.

The image lattice  $\Lambda$  is divided into overlapping patches for multiple scales by a scanning window with a step size about 10% of the window size. Then we calculate their corresponding prototypes and responses for all images in  $D^+$ . The sketch prototypes  $B_i$  are specified by the Gabor dictionary  $\Delta$ , and the histogram prototypes  $\mathbf{h}_k$  are obtained by computing the histograms for positive examples in the same region of template lattice and then taking the average. As a result, we obtain an excessive number of candidate patches.

$$\Omega_{\text{cand}} = \{\Lambda_j, \ell_j, \{B_j \text{ or } \mathbf{h}_j\} : j = 1, 2, \dots, M\}. \quad (6.28)$$

From  $\Omega_{\text{cand}}$ , we will select the most informative patches and their corresponding prototypes for HIT.

By induction, at the  $k$ -th step, we have a HIT with  $k - 1$  patches and a model  $p = p_{k-1}$ :



$$\text{HIT}_{k-1} = (\{\Lambda_j, \ell_j, B_j \text{ or } \mathbf{h}_j, \delta_j, j = 1, \dots, k-1\}, \Theta_{k-1}).$$

Consider a new candidate patch  $\Lambda_k$  in  $\Omega_{\text{cand}}$  and its responses on  $n$  positive examples and  $N$  negative examples:

$$\{r_{k,i}^+, i = 1, \dots, n\} \quad \{r_{k,i}^-, i = 1, \dots, N\}. \quad (6.29)$$

And let  $\bar{r}_k^+$  and  $\bar{r}_k^-$  be the sample means on the two sets.

The gain of adding this patch to the template is measured by the KL divergence between the target marginal distribution  $f(r_k)$  and the current model  $p_{k-1}(r_k)$ , as this represents the new information in the training data that is not yet captured in the model. Among all the candidate patches, the one with the largest gain is selected.

To estimate this gain, we use Monte-Carlo methods with samples from  $f(r_k)$  and  $p_{k-1}(r_k)$ . Obviously  $\{r_{k,i}^+\}$  is a fair sample from  $f(r_k)$ . While to sample from  $p_{k-1}(r_k)$ , one may use importance sampling on  $\{r_{k,i}^-\}$ , *i.e.* re-weighting the examples by  $\frac{p_{k-1}(r_k)}{q(r_k)}$ . Here we simplify the problem by a conditional independence assumption as stated in previous section. A feature response  $r_1(\mathbf{I}_{\Lambda_1})$  is roughly uncorrelated with  $r_2(\mathbf{I}_{\Lambda_2})$  if one of the following holds: i) the two patches  $\Lambda_1$  and  $\Lambda_2$  have little overlap; ii)  $\Lambda_1$  and  $\Lambda_2$  are from different scales. If at the  $k$ -th step we have removed from  $\Omega_{\text{cand}}$  all the candidate patches that overlap with selected patches, then  $r_k$  is roughly uncorrelated with all the previously selected responses  $r_1, \dots, r_{k-1}$ . As a result,  $p_{k-1}(r_k) = q(r_k)$  and  $\{r_{k,i}^-\}$  can be used as a sample of  $p_{k-1}(r_k)$ . The exact formula for estimating the gain (*i.e.* KL divergence between  $f(r_k)$  and  $p_{k-1}(r_k)$ ) is given, once we have derived the parametric form of  $p$  in the following.

For a selected patch  $\Lambda_k$ , the new model  $p = p_k$  is required to match certain observed statistics (*e.g.* first moment) while it should be also close to the learned model  $p_{k-1}$  to preserve the previous constraints.

$$p_k^* = \arg \min \quad \text{KL}(p_k | p_{k-1}) \quad (6.30)$$

$$s.t. \quad E_{p_k}[r_k] = E_f[r_k] \quad (6.31)$$

By solving the Euler-Lagrange equation with Lagrange multipliers  $\{\lambda_j\}$  and  $\gamma$ ,

$$\begin{aligned} \frac{\partial}{\partial p_k} \left\{ \sum_{\mathbf{I}} p_k(\mathbf{I}) \log \frac{p_k(\mathbf{I})}{p_{k-1}(\mathbf{I})} + \lambda_k (E_{p_k}[r_k] - E_f[r_k]) \right. \\ \left. + \gamma (\sum_{\mathbf{I}} p_k(\mathbf{I}) - 1) \right\} = 0. \end{aligned}$$

Thus we have,

$$p_k(\mathbf{I}) = p_{k-1}(\mathbf{I}) \frac{1}{z_k} \exp\{-\lambda_k r_k(\mathbf{I})\}. \quad (6.32)$$

$z_k = E_q[\exp\{\lambda_k r_k(\mathbf{I}_{\Lambda_k})\}]$  is a normalizing constant. This can be estimated by the negative samples,

$$z_k \approx \frac{1}{N} \sum_{i=1}^N e^{\lambda_k r(\mathbf{J}_{i, \Lambda_k})}. \quad (6.33)$$

$\lambda_k$  is the parameter (Lagrange multiplier) to satisfy constraint in eqn (6.31),

$$E_p[r_k] \approx \frac{1}{N} \sum_{i=1}^N \left[ r(\mathbf{J}_{i, \Lambda_k}) e^{\lambda_k r(\mathbf{J}_{i, \Lambda_k})} \right] \frac{1}{z_k} = \bar{r}_k^+. \quad (6.34)$$

In computation, we can look up  $\bar{r}_k^+$  in the table to find the best  $\lambda_k$ . The importance sampling is a good estimation in calculating  $\lambda_k$  and  $z_k$  because in our model  $r$  is one dimensional.

By recursion, we have a *factorized* log-linear form,

$$p_K(\mathbf{I}) = q(\mathbf{I}) \prod_{j=1}^K \left[ \frac{1}{z_j} \exp\{\lambda_j r_j(\mathbf{I}_{\Lambda_j})\} \right]. \quad (6.35)$$

### 6.3.5 Example: vector fields for human hair analysis and synthesis

In this subsection, we demonstrate an example of applying the primal sketch model to human hair analysis and synthesis [24]. Specifically, the hair images can be treated as 2D piecewise smooth vector (flow) fields, and thus the representation is view-based in contrast to the physically-based 3D hair models in graphics. The primal sketch model has three levels. The bottom level is the high-frequency band of the hair image. The middle level is a piecewise smooth vector field for the hair orientation, gradient strength, and growth directions. The top-level is an attribute sketch graph for representing the discontinuities in the vector field. Besides the three level representation, the shading effects, i.e. the low-frequency band of the hair image, are modeled by a linear superposition of some Gaussian image bases, and the hair color is encoded by a color map. Figure 6.9 shows an example.

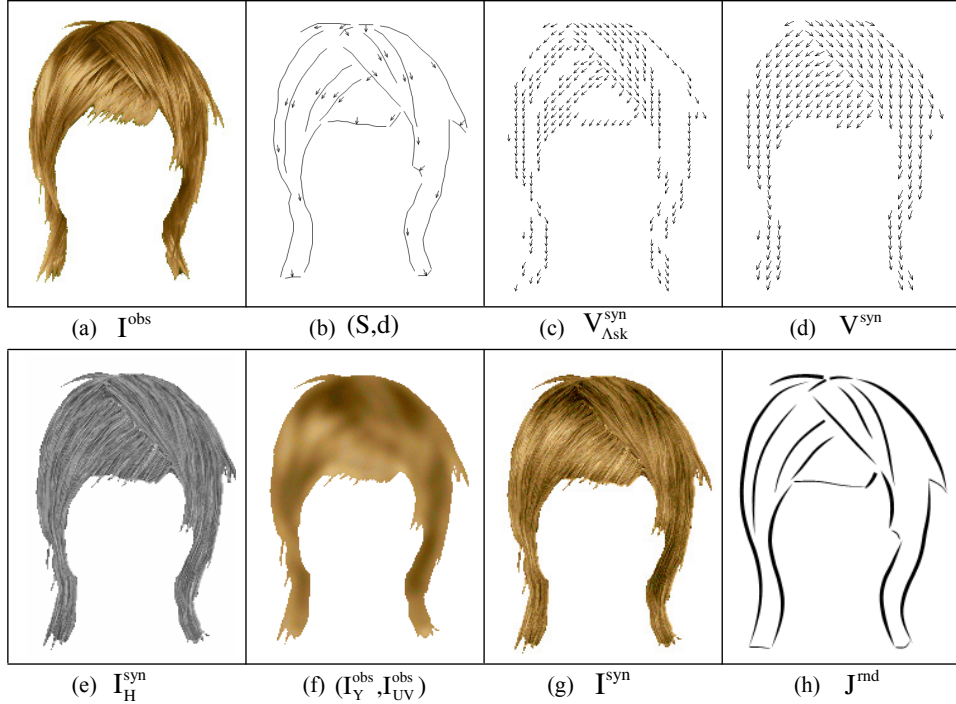


Figure 6.9: Reprinted with permission from [24]. Example of hair model and inference. (a) is an input color image  $\mathbf{I}^{\text{obs}}$ , (b) is the computed sketch  $\mathbf{S}$  with directions  $\mathbf{d}_S$ . (c) is the sketchable vector field  $\mathbf{V}_{\Lambda_{sk}}$  generated from  $(\mathbf{S}, \mathbf{d}_S)$ . (d) is the overall vector field  $\mathbf{V}$  after filling-in non-sketchable part. (e) is the high frequency hair texture image  $\mathbf{I}_H^{\text{syn}}$  generated from the vector field. (f) is the shading and lighting image. (g) is the synthesized color image  $\mathbf{I}^{\text{syn}}$  after adding the shading and color. We render an artistic sketch  $\mathbf{J}^{\text{rnd}}$  in (h).

Let  $\mathbf{I}^{\text{obs}}$  denote an observed color hair image. By a Luv transform, we obtain an intensity image  $\mathbf{I}_Y^{\text{obs}}$  and a color channel image  $\mathbf{I}_{UV}^{\text{obs}}$ . The color channel  $\mathbf{I}_{UV}^{\text{obs}}$  is discretized into a small number of colors and represented by a color map from the intensity  $[0, 255]$  of  $\mathbf{I}_Y^{\text{obs}}$  to a color. The intensity image  $\mathbf{I}_Y^{\text{obs}}$  is further decomposed into a low frequency band  $\mathbf{I}_L^{\text{obs}}$  for illumination and shading with a low-pass Gaussian filter, and the remaining high frequency band is the texture for the hair pattern  $\mathbf{I}_H^{\text{obs}}$ . The low frequency band is simply represented by a linear superposition of Gaussian image bases plus a mean intensity  $\mu$ ,

$$\mathbf{I}_L^{\text{obs}}(x, y) = \mu + \sum_{i=1}^{K_L} \alpha_i G(x - x_i, y - y_i; \theta_i, \sigma_{xi}, \sigma_{yi}) + \text{noise}. \quad (6.36)$$

Usually  $K_L = O(10)$ . Each Gaussian base is represented symbolically by an ellipse for editing and it has five parameters for the center, orientation, and standard deviation along the two axes. The coefficients  $\{\alpha_i\}$  can be positive or negative for highlights and shadows respectively. The Matching Pursuit algorithm is used to automatically extract the coefficients from the input image.

Our study is focused on the texture appearance  $\mathbf{I}_H^{\text{obs}}$  with a three level primal sketch model. A hair texture  $\mathbf{I}_H$  on a lattice  $\Lambda$  is generated by a hidden layer  $\mathbf{V}$  – the vector field for hair growth flow, and  $\mathbf{V}$  is in turn generated by an attribute hair sketch  $\mathbf{S}$  which is a number of sketch curves representing the boundaries of hair strands and wisps with direction  $\mathbf{d}_S$ .

$$\text{Sketch } (\mathbf{S}, \mathbf{d}_S) \xrightarrow{\Delta_{\text{sk}}} \text{Vector field } \mathbf{V} \longrightarrow \text{hair image } \mathbf{I}_H$$

$\Delta_{\text{sk}}$  is a dictionary of sketch primitives shown in Fig. 6.10. Each primitive is a rectangular window (say  $5 \times 7$  pixels) and some examples are shown in Fig. 6.11.

Primitives	(a)	(b)	(c)	(d)	(e)
orientation fields					
vector fields with $(d_l, d_r)$	 (1,1)	 (1,1)	 (1,1) (1,-1)	 (1,1)	 (1,1)
	 (1,-1)	 (1,-1)	 (-1,1) (-1,-1)		 (-1,-1)

Figure 6.10: Reprinted with permission from [24]. Five primitives for the orientation field, and eleven primitives for the directed vector field  $\mathbf{V}$  in a dictionary  $\Delta_{\text{sk}}$ . (a) Side boundary. (b) Source (origin) or sink (end) of hair strands. (c) Occluding boundary. (d) Dividing line. (e) Stream line. The line segments and arrows in the primitive windows show the canonical orientations, and the angles may change in  $[-\pi/6, \pi/6]$ .

For synthesizing hair images, we assume that  $(\mathbf{S}, \mathbf{d}_S)$  is either inferred from a hair image or edited manually through a simple user interface. From  $(\mathbf{S}, \mathbf{d}_S)$ , we synthesize a hair image  $\mathbf{I}_H^{\text{syn}}$  in three steps according to the generative model.

1. Synthesizing the vector field from the sketch  $\mathbf{V}^{\text{syn}} \sim p(\mathbf{V}|\mathbf{S}, \mathbf{d})$ .
2. Synthesizing the hair texture from the vector field  $\mathbf{I}_H^{\text{syn}} \sim p(\mathbf{I}_H|\mathbf{V}^{\text{syn}})$ .
3. Synthesizing color image  $\mathbf{I}^{\text{syn}}$  by adding a shading image  $\mathbf{I}_L^{\text{syn}}$  to  $\mathbf{I}_H^{\text{syn}}$  and then transfer the grey image to color by the color map. Fig. 6.12 shows three examples of hair synthesis.

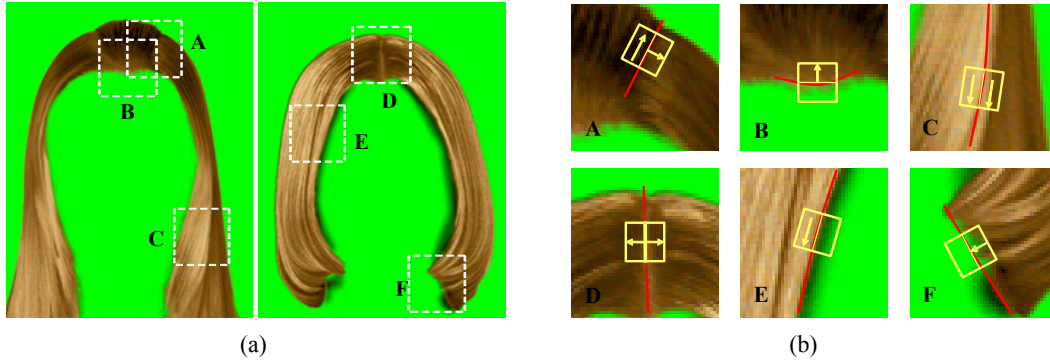


Figure 6.11: Reprinted with permission from [24]. (a) Windows A-F are 6 primitive examples. (b) Zoomed-in views of the six windows.

The inference algorithm is divided into two stages: (i) compute the undirected orientation field and sketch graph from an input image, and (ii) compute the hair grow direction for the sketch curves and the orientation field using a Swendsen-Wang cut algorithm. Both steps maximize a joint Bayesian posterior probability. This generative primal sketch model provides a straightforward way for synthesizing realistic hair images and stylistic drawings (rendering) from a sketch graph and a few Gaussian bases.

## 6.4 Relations between Primal Sketch and the HoG and SIFT Representations

In this section, we will draw some connections between the primal sketch and Histogram of Oriented Gradients (HOG) [33] and Scale-Invariant Feature Transfer (SIFT) [143, 144] representations, which was widely adopted in computer vision as generic image features and object templates before deep neural networks.

Histogram of Oriented Gradient (HOG) [33] is based on evaluating well-normalized local histograms of image gradient orientations in a dense grid. The basic idea is that local object appearance and shape can often be characterized rather well by the distribution of local intensity gradients or edge directions, even without precise knowledge of the corresponding gradient or edge positions. In practice this is implemented by dividing the image window into small spatial regions ("*cells*"), for each cell accumulating a local 1-D histogram of gradient directions or edge orientations over the pixels of the cell (Figure 6.13). The combined histogram entries form the representation. For better invariance to illumination, shadowing, etc., it is also useful to contrast-normalize the local responses before using them. This can be done by accumulating a measure of local histogram "energy" over somewhat larger spatial regions ("*blocks*") and using the results to normalize all of the cells in the block. The normalized descriptor blocks are usually referred to as Histogram of Oriented Gradient (HOG) descriptors. A typical application of HOG descriptors is human detection, which is achieved by tilting the detection window with a dense grid of HOG descriptors and using the combined feature vector in a conventional SVM-based window classifier.

The connection between HOG and the primal sketch is as follows. If the cell used for the HOG descriptor happens to be a fine-scale stochastic texture (like the sky, wall, clothes, shading patch, with no visible sketchable structures), then keeping the histogram of gradient corresponds to the FRAME model for texture in the primal sketch model. If the cell has visible structures, and is thus sketchable, its HOG histogram will be highly peaked and thus dominated by 1-2 bins. For this case, we adopt an explicit representation by the sketches (structure) in the primal sketch model.

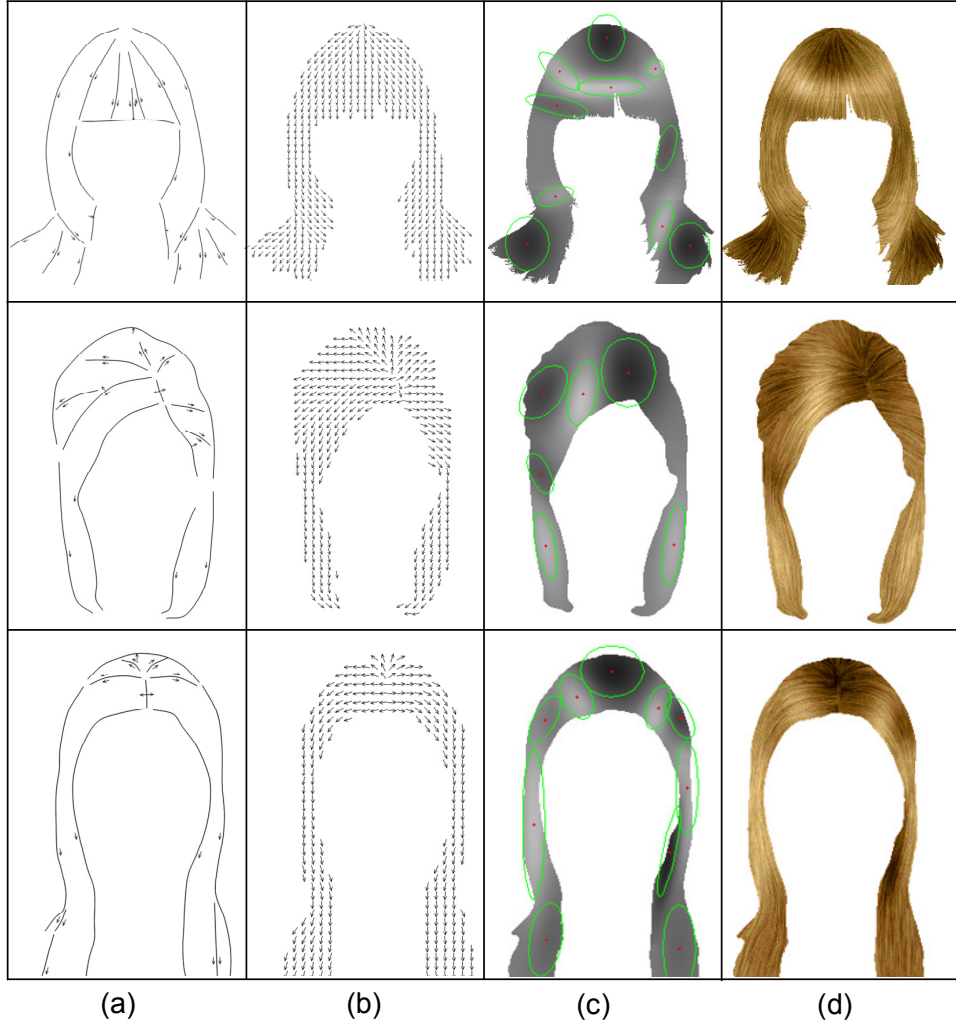


Figure 6.12: Reprinted with permission from [24]. Three examples of hair drawing and synthesis. (a) Manually input hair sketch  $\mathbf{S}$  with directions  $\mathbf{d}_S$ . (b) Synthesized vector field  $\mathbf{V}^{\text{syn}}$  given  $(\mathbf{S}, \mathbf{d}_S)$ . (c) Edited shading maps with a small number of ellipses. (d) Synthesized color images  $\mathbf{I}^{\text{syn}}$ .

Scale-Invariant Feature Transform (SIFT) [144] transforms an image into a large collection of local feature vectors, each of which is invariant to image translation, scaling, and rotation, and partially invariant to illumination changes and affine or 3D projection. It consists of four major stages: (1) scale-space peak selection; (2) keypoint localization; (3) orientation assignment; (4) keypoint descriptor. In the first stage, potential interest points are identified by scanning the image over location and scale. This is implemented efficiently by constructing a Gaussian pyramid and searching for local peaks (termed keypoints) in a series of difference-of-Gaussian (DoG) images. In the second stage, candidate keypoints are localized to sub-pixel accuracy and eliminated if found to be unstable. The third stage identifies the dominant orientations for each keypoint based on its local image patch. The assigned orientation(s), scale, and location for each keypoint enable SIFT to construct a canonical view for the keypoint that is invariant to similarity transforms. The final stage builds a local image descriptor for each keypoint, based on a patch of pixels in its local neighborhood.

SIFT can be applied to object recognition, which is performed by first matching each keypoint independently to the database of keypoints extracted from training images. Many of these initial matches will be

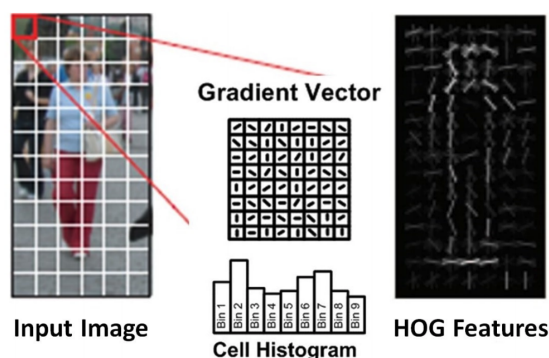


Figure 6.13: Feature extraction with histogram of oriented gradients (HOG) descriptors. The image window is first divided into small spatial regions ("cells"), and then for each cell a local 1-D histogram of gradient directions or edge orientations over the pixels of the cell is accumulated.

incorrect due to ambiguous features or features that arise from background clutter. Therefore, clusters of at least 3 features are first identified that agree on an object and its pose, as these clusters have a much higher probability of being correct than individual feature matches. Then, each cluster is checked by performing a detailed geometric fit to the model, and the result is used to accept or reject the interpretation. Figure 6.14 shows an example of object recognition using SIFT for a cluttered and occluded image containing 3D objects. The training images of a toy train and a frog are shown on the left. The middle image contains instances of these objects hidden behind others and with extensive background clutter so that detection of the objects may not be immediate even for human vision. The image on the right shows the final correct identification superimposed on a reduced contrast version of the image. The keypoints that were used for recognition are shown as squares with an extra line to indicate orientation. The sizes of the squares correspond to the image regions used to construct the descriptor. An outer parallelogram is also drawn around each instance of recognition, with its sides corresponding to the boundaries of the training images projected under the final affine transformation determined during recognition.

The SIFT keys can be considered an extension to the keypoints in images, like color or complex object patterns. It corresponds to a local sub-graph in the primal sketch, with some scale invariance.



Figure 6.14: Reprinted with permission from [144]. The training images for two objects are shown on the left. These can be recognized in a cluttered image with extensive occlusion, shown in the middle. The results of recognition are shown on the right. A parallelogram is drawn around each recognized object showing the boundaries of the original training image under the affine transformation solved for during recognition. Smaller squares indicate the keypoints that were used for recognition.





## 2.1D Sketch and Layered Representation

Primal sketch is a generic two-layer 2D representation in terms of how image content is explained away with respect to either explicit functions of basis functions or implicit functions of feature statistics. Primal sketch seeks to decompose an image domain into the texton structural domain and the remaining texture domain. In this chapter, we shall study how to decompose an input image into multiple layers with partial occluding order relation inferred and the occluded contour completed if possible (Fig. 7.1). In the pioneering work, the resulting representation is called *the 2.1D sketch*, by Nitzberg and Mumford [169], to bridge low-level image primitives (including textons such as edges and junctions and texture atomic regions) and middle-level Marr's 2.5D sketch (to be presented in the next chapter), and termed *layered image representation*, by Adelson and Wang [2, 223, 224], in image coding and motion analysis.

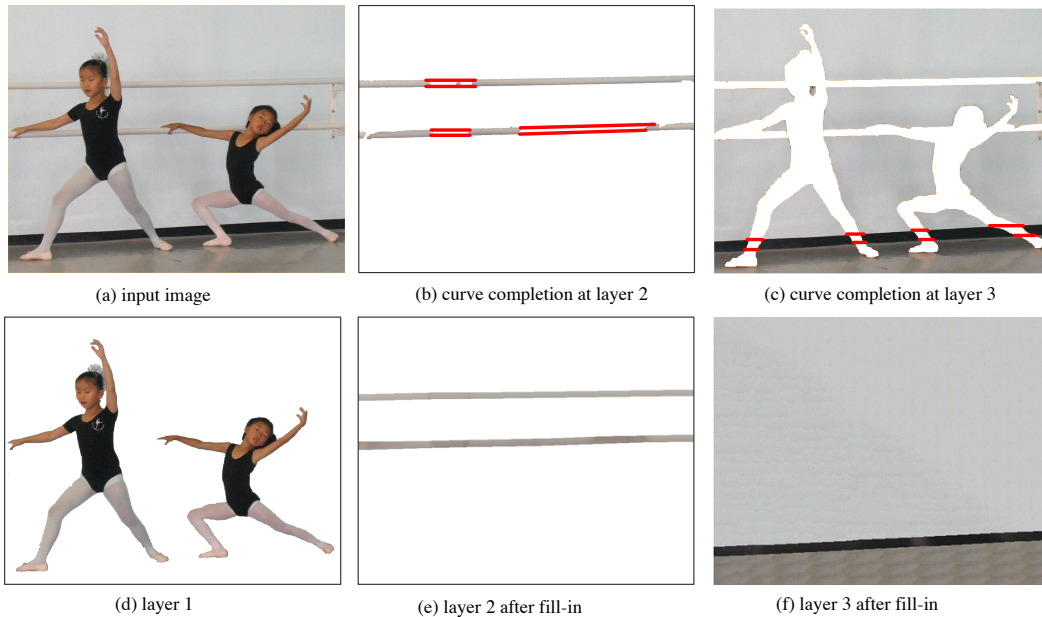


Figure 7.1: Illustration of the 2.1D sketch and layered image representation. From a 2D sketch to a 2.1D layered representation by reconfiguring the bond relations between regions in different layers. (a) is an input image from which a 2D primal sketch is computed. This is transferred to a 2.1D sketch representations with three layers shown in (d), (e) and (f) respectively. The inference process reconfigures the bonds of the image primitives shown in red in (b) and (c).

In practice, the idea of layered representation has been widely used for image manipulation in image editing software such as Adobe’s Photoshop (for example, adding text to an image, or adding vector graphic shapes, or applying a layer style to add a special effect such as a drop shadow or a glow). In the literature of computer vision, the 2.1D sketch or layered image representation has also been studied from other perspectives including line drawing interpretations [149,221], segmentation [50,212], occlusion recovery [193,229], contour illusory and completion [48,49,95,183,222], and figure-ground separation [184,206].

The 2.1D sketch stems from the figure-ground separation/organization problem, a type of perceptual grouping that contributed to visual object recognition. In Gestalt psychology, the figure-ground separation problem is usually posed as identifying a *figure* (such as the two girls in Fig. 7.1) from the *background* (such as the dancing practice room in Fig. 7.1). Computing the figure-ground organization of an input image can help resolve perceptual ambiguities in for example the face-vase and martini-bikini drawings. The 2.1D sketch captures the partial occluding order between multiple object surfaces/generic regions in a scene, representing the rank information of relative depth among them, and thus providing a critical representation scheme for addressing the multistable perception phenomenon in vision.

In this chapter, we focus on the problem of computing the 2.1D sketch from a monocular image under the variational formulation framework, the energy minimization framework, and the Bayesian inference framework respectively.

## 7.1 Problem Formulation

In this section, we introduce the notation and present a high-level formulation. We will elaborate on different components in sections followed.

Denote by  $\mathcal{D}$  an image domain and  $\mathbf{I}$  an image defined on the domain  $\mathcal{D}$ . To infer the 2.1D sketch, we will build up a three-layer model from the input image  $\mathbf{I}$ , to its 2D representation, denoted by  $W_{2D}$  and to the 2.1D sketch, denoted by  $W_{2.1D}$ . We have,

$$\mathbf{I} \Rightarrow W_{2D} \Rightarrow W_{2.1D}. \quad (7.1)$$

$W_{2D}$  represents a set of 2D elements to be layered, which can be 2D atomic regions or 2D curves and curve groups, as well as the associated attributes if needed in inference. For example, if we only consider 2D regions, we have,

$$W_{2D} = \{n, (R_i, l_i, \theta_i)_{i=1}^n\}, \quad (7.2)$$

which consists of  $n$  regions,  $R_i$ , each of which is represented by a region model indexed by  $l_i$  in a predefined model family with parameters  $\theta_i$ .

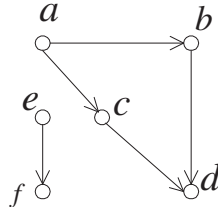


Figure 7.2: Reprinted with permission from [215]. A Hasse diagram for a partial order relation

$W_{2.1D}$  represents a set of surfaces, their partial occluding order and contours used to complete regions in different surfaces. A surface consists of one 2D region or more than one 2D region with completed contours.

Any 2D region in  $W_{2D}$  belongs to one and only one surface in the layered representation. We have,

$$W_{2.1D} = \{m, \{S_i\}_{i=1}^m, \mathcal{PR}\} \quad (7.3)$$

which consists of  $m$  surfaces ( $m \leq n$ ) and  $\cup_i S_i = \mathcal{D}$ .  $\mathcal{PR}$  represents the partial occluding order between the set of  $m$  surfaces. Consider a set  $A = \{a, b, c, d, e, f\}$ , we define a partially ordered set, *poset* [203],  $\mathcal{PR} = \langle A, \preceq \rangle$ .  $b \preceq c$  means that surface  $b$  occludes surface  $c$  or  $b$  is on top of  $c$ .  $\mathcal{PR}$  is represented by a directed acyclic graph (DAG) called a Hasse diagram. Figure 7.2 shows an example of the Hasse diagram for  $\mathcal{PR} = \{\langle a, b \rangle, \langle b, d \rangle, \langle a, d \rangle, \langle a, c \rangle, \langle c, d \rangle, \langle e, f \rangle\}$  on the set  $A$ .

Denote the “visible” portion of a surface  $S_i$  by,

$$S'_i = S_i \setminus \cup_{S_j \preceq S_i} S_j. \quad (7.4)$$

Then, the recovered curve(s) in the surface  $S_i$  can be defined by,

$$C_i = \partial S_i \setminus \partial S'_i. \quad (7.5)$$

Some  $C_i$ ’s may be empty. For example, the front most surface does not have occluded parts and the occluded parts in the background are usually left open.

To infer the 2.1D sketch, we will study two different formulations which cast the inference under the energy minimization framework and under the Bayesian framework respectively. In either case, there are two different assumptions:

- Assume the 2D representation  $W_{2D}$  has been computed already and will be fixed in the inference of the 2.1D sketch. This is typically adopted in the literature.
- Compute the 2D representation  $W_{2D}$  and the 2.1D sketch  $W_{2.1D}$  jointly. This leads to a much larger search space and usually entails more powerful search algorithms such as DDMCMC [214].

## 7.2 The Variational Formulation by Nitzberg and Mumford

In their seminal work [169], Nitzberg and Mumford proposed a variational formulation for inferring the 2.1D sketch extended from the Mumford-Shah energy functional [162]. This belongs to the energy minimization framework and we briefly introduce the formulation in this section.

### 7.2.1 The Energy Functional

We first overview the Mumford-Shah energy functional for low-level image segmentation. It is a piece-wise smooth model that aims to segment an image into as few and simple regions as possible while keeping the color of each region as smooth and/or slowly varying as possible. The functional is defined such that it takes its minimum at an optimal piece-wise smooth approximation to a given image  $I$  defined on the domain  $\mathcal{D}$ . An approximation function, denoted by  $f$ , is smooth except at a finite set  $\Gamma$  of piece-wise contours which meet the boundary of the image domain,  $\partial\mathcal{D}$ , and meet each other only at their endpoints. So, the contours of  $\Gamma$  segments the image domain into a finite set of disjoint regions, denoted by  $R_1, \dots, R_n$ , *i.e.*, the connected components of  $\mathcal{D} \setminus \Gamma$ . The Mumford-Shah functional is defined to measure the match between an image  $I$  and a segmentation  $f, \Gamma$ :

$$\mathcal{E}_{M-S}(f, \Gamma|I) = \mu^2 \int_{\mathcal{D}} (f - I)^2 dx + \int_{\mathcal{D} \setminus \Gamma} \|\nabla f\|^2 dx + v \int_{\Gamma} ds \quad (7.6)$$

where in the right hand side the first term measures how good  $f$  approximates  $I$ , the second asks that  $f$  vary slowly except at boundaries, and the third asks the set of contours be as short, and hence as simple and straight as possible.

Similar in spirit to the Mumford-Shah functional, Nitzberg and Mumford proposed an energy functional that achieves a minimum at the optimal overlapping layering of surfaces. For simplicity, assume  $S_i$  is a closed subset of  $\mathcal{D}$  with piece-wise smooth boundary and connected interior. The energy functional is defined by:

$$\mathcal{E}_{2.1D}(\{S_i\}, \mathcal{PR}|I) = \sum_{i=1}^n \left\{ \mu^2 \int_{S'_i} (I - m_i) dx + \epsilon \int_{S_i} dx + \int_{\partial S_i \setminus \partial \mathcal{D}} \phi(\kappa) ds \right\} \quad (7.7)$$

where  $m_i$  is the mean of the image  $I$  on  $S'_i$  and  $\kappa$  is the curvature of the boundary  $\partial S_i$ . The function  $\phi : \mathbf{R} \rightarrow \mathbf{R}$  is defined by,

$$\phi(\kappa) = \begin{cases} v + \alpha \kappa^2, & \text{for } |\kappa| < \beta/\alpha \\ v + \beta |\kappa|, & \text{for } |\kappa| \geq \beta/\alpha \end{cases} \quad (7.8)$$

The functional above can only address inferring the 2.1D sketch from simple input images, which is a reasonable assumption in the early day of computer vision. First, the model does not allow self-overlapping “woven” surfaces such as what a garden hose would project, nor folded surfaces such as that produced by an image of a sleeve whose edge disappears around the back of an arm. Second, the model is piecewise constant with a constant mean being used for the surface model. Third, transparency and shadows are not handled. We will study more expressive models with more powerful inference algorithms for handling those issues frequently observed in modern real images.

### 7.2.2 The Euler Elastica for Completing Occluded Curves

To recover the occluded curves, Nitzberg and Mumford adopted the Euler Elastica method to interpolate them. Suppose a surface  $S_i$  disappears behind occluding objects at a point  $P_0 \in \partial S_i$  and reappears at  $P_1$ . Let  $t_0$  and  $t_1$  be the unit tangent vectors to  $\partial S_i$  at  $P_0$  and  $P_1$  respectively. Then, computing the occluded curve  $C_i$  between  $P_0$  and  $P_1$  is posed as a minimization problem:

$$\mathcal{E}(C_i|S_i, P_0, t_0, P_1, t_1) = \int_{C_i} (v + \alpha \kappa^2) ds \quad (7.9)$$

Computationally, the simplest way to solve the Elastic seems to be hill-climbing. It starts with a convenient chain  $x_0 = P_0, x_1, \dots, x_N = P_1$  of points for which  $x_1 - x_0 \propto t_0$  and  $x_{N-1} - x_N \propto t_1$ .  $X_i$ 's ( $i = 1, \dots, N-1$ ) are computed by letting them evolve to decrease the Elastica functional.

## 7.3 The Mixed Markov Random Field Formulation

In this section, we address the 2.1D sketch problem with the mixed Markov Random Field (MRF) representation and under the Bayesian inference framework. We consider 2D image regions only as layering primitives. As illustrated in Figure 7.3, given the set of segmented regions (in different colors) of an input image, our objective consists of two components: **region layering or coloring** to divide the set of regions into an unknown number of layers, and **contour completion** to recover the occluded contours between regions in the same layer. In addition, we will also fit the probability models of regions in different layers and preserve multiple distinct interpretations accounting for the intrinsic ambiguities if needed. Two key observations in modeling and computing the 2.1D sketch problem are as follows.

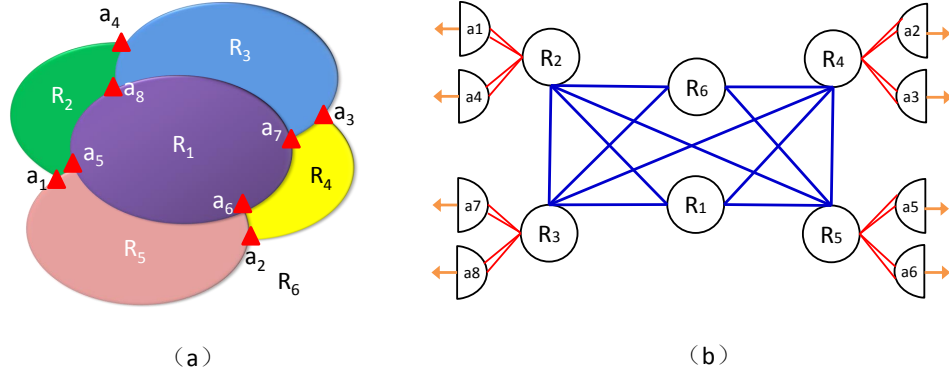


Figure 7.3: A simple example of the 2.1D sketch and the mixed MRF modeling. (a) is a set of 2D regions to be layered, (b) is the initial reconfigurable graph with mixed MRF including two types of nodes, region nodes and terminator nodes. See text for details.

i) *Long range interactions* between 2D regions. In order to determine which 2D regions are in the same layer and to complete the contour of occluded parts, long-range interactions are entailed, which imposes different requirements in modeling than traditional local relation modeling using Markov Random Field (MRF) models. Consider region 3 and 5 in Figure 7.1. Whether they should merge into a single surface or should remain independent in the same layer depends on their compatibility across long-range interactions.

ii) *Dynamic neighborhood system*. In the contour completion process, given a current layering assignment, the neighborhood of any end point on an occluded contour looking for its corresponding point is constrained to match among the set of points in the same layer. This means the neighborhood system changes according to different layering assignments at each step in the inference, which is different from the fixed neighborhood system in the traditional MRF.

The two properties stated above lead traditional MRFs to fail to model the 2.1D sketch problem and call for new methods, such as the mixed Markov random field [57, 58] to be used in this chapter. Both MRF and mixed MRF can be presented by graphical models. A mixed MRF differs from traditional MRFs in its definition of its neighborhood system, which is static in the latter, but is dynamic in the former due to the introduction of a new type of nodes in the graph. Concretely speaking, a mixed MRF has the following two characteristics:

- i) Nodes are inhomogeneous with different degrees of connections, which are inferred from images on-the-fly.
- ii) The neighborhood of each node is no longer fixed but inferred as open bonds or address variables, which reconfigures the graph structure.

Following Eqn. 7.1, we adopt a three-layer generative image model consisting of the input image, the 2D representation, and the 2.1D sketch. We use 3-degree-junctions such as T-junctions, Y-junctions, or arrow junctions as cues for occlusion. The 2D representation consists of a set of 2D regions (see  $R_1, R_2, \dots, R_6$  in Figure 7.3 (a)) and a set of terminators (see  $a_1, a_2, \dots, a_8$ ) broken from detected T-, Y-, or arrow junctions during the process of layering and, if possible, to be completed in the inferred layered representation. So, in the graphical representation (see Figure 7.3 (b)), there are two types of nodes: one is the region nodes and the other terminator nodes. Region nodes constitute the region adjacency graph to be partitioned during the

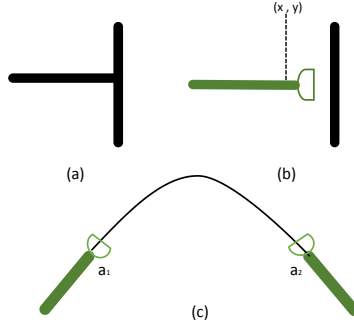


Figure 7.4: The open bounds or address variable. A T-junction is shown in (a), it is broken into a terminator in (b) as an open bound described as an address variable including location  $(x, y)$ , the orientation of the terminator, and appearance information inherited from the region it belongs to; (c) is the Elastica computed using two terminators in the process of contour completion.

process of layering. The terminator nodes make the graph reconfigurable since the neighborhood system of terminator nodes is not static and depends on the current assignment of layering. For such reconfigurable graphical models, it has been shown that a probability model defined on them still observes a suitable form of the Hammersley-Clifford theorem and can be simulated by Gibbs sampling [57].

### 7.3.1 Definition of $W_{2D}$ and $W_{2.1D}$

For the clarity of the presentation, we mainly focus on inferring the 2.1D sketch with the 2D representation given and fixed.

**The 2D representation.**  $W_{2D}$  consists of a set of 2D regions,  $V_R$ , and a set of 3-degree-junctions,  $V_T$ , such as T-, Y-, or arrow junctions:

$$W_{2D} = (V_R, V_T) \quad (7.10)$$

$$V_R = (R_1, R_2, \dots, R_N) \quad (7.11)$$

$$V_T = (T_1, T_2, \dots, T_M) \quad (7.12)$$

$V_R$  is the regions set, which can be many atomic regions or composed of a few image patches.  $V_T$  consists of those junctions selected from the detected 3-degree-junctions [230] that are assigned to the corresponding regions.

**Open bonds or address variables.** 3-degree-junctions will be broken into a set of terminators, illustrated in Fig. 7.4 (a) and (b) for a T-junction, as the open bonds, or address variables,  $V_B$ , of corresponding regions during the inference. The open bonds are like pointers to regions, initially kept open but to be assigned an address variable and completed during inference. In practice, each bond has a set of attributes,  $A_B$ , including those belonging to itself and those inherited from the region. These attributes often include geometric transformation features, such as location, orientation, and, length, and some appearance features such as features computed from region models. They are used to test the compatibility of any two bonds, deciding whether or not to link together. Often, besides the foreground and background regions, each region has two or more (assumed no more than  $m \leq M$ ) open bonds with their ownership defined, denoted as  $B(R_i), \forall i \leq N$ .

$$V_B = (a_1, a_2, \dots, a_M) \quad (7.13)$$

$$A_B = (A_{a1}, A_{a2}, \dots, A_{aM}), a_i \in V_B, \forall i \leq M \quad (7.14)$$

$$B(R_i) = (a_{i1}, a_{i2}, \dots, a_{im}), a_{ij} \in V_B, \forall i \leq N, \forall j \leq m \quad (7.15)$$

**Generative models of regions.** We adopt generative models for each region. Let  $R \subset \Lambda$  denote a 2D region and  $\mathbf{I}_R$  the intensities in  $R$  or color values in  $(r, g, b)$  color space. The model assumes constant intensity or color value with additive noise  $\eta$  modeled by a non-parametric histogram  $H$ . The model can be learned off-line and more sophisticated models can easily be added to the algorithm.

$$J(x, y, \theta) = \mu, \mathbf{I}_R(x, y) = J(x, y, ; \theta) + \eta, \eta \sim H \quad (7.16)$$

With a slight abuse of notation, we denote the parameters used in a region by  $\theta = (\mu, H)$ . Here  $\mu$  is the mean of a region  $R$  or a connected component  $CP$ . The likelihood probability is

$$p(\mathbf{I}_R | R, \theta) \propto \prod_{(x, y \in R)} H(\mathbf{I}_R(x, y) - J(x, y, \theta)) \quad (7.17)$$

The prior for a region  $R$  assumes short boundary length  $\partial R$ . This is to encourage smoothness, and a compact area  $\|R\|$ ,

$$p(R) \propto \exp\{-\gamma_r |R|^\rho - \frac{1}{2} \lambda |\partial R|\} \quad (7.18)$$

where  $\rho$  and  $\lambda$  are fixed constants and  $\gamma$  is a fixed scale factor for regions.

**Terminator representation and the Elastica.** When a terminator  $a_i$  is broken from a junction, it is represented by its attributes  $A_{ai} = (x, y; ori, \kappa, p_f)$ , where  $(x, y)$  is location,  $ori$  the orientation of the terminator,  $\kappa$  the curvature, and  $p_f$  the profile or some attributes inherited from the region.

Given two terminators,  $a_i$  and  $a_j$ , the contour to be completed between them, denoted by  $\Gamma^*$ , is decided by minimizing the Elastica cost function in a contour space  $\Omega_\Gamma$  [100, 119, 162, 169]

$$\begin{aligned} \Gamma^* &= \arg \min_{\Gamma \in \Omega_\Gamma} \mathcal{E}(\Gamma; a_i, a_j) \\ &= \arg \min_{\Gamma \in \Omega_\Gamma} \int_\Gamma [(\nu_1 + \alpha_1 \kappa_1^2) + (\nu_2 + \alpha_2 \kappa_2^2)] ds \end{aligned} \quad (7.19)$$

where  $\kappa_1$  is the curvature of  $a_1$ ,  $\kappa_2$  is the curvature of  $a_2$ ,  $\nu_1$  and  $\nu_2$  are constants, and  $\alpha_1$  and  $\alpha_2$  are scalable coefficients. The parameters  $\nu_1, \nu_2, \alpha_1$  and  $\alpha_2$  are learned from an image data set [253].

**The 2.1D sketch.**  $W_{2.1D}$  is represented by a set of labels,  $X$ , for the layer information of regions and a set of variables,  $Y$ , describing address variable assignments. Both of them are inferred from the image. In addition, according to the layering labels and assignments of address variables, a set of surfaces,  $S_f$ , is formed which consists of one or more regions merged through a set of recovered contours,  $C_t$ .

$$W_{2.1D} = (X, Y; S_f, C_t) \quad (7.20)$$

$$X = (x_{R1}, x_{R2}, \dots, x_{RN}) \quad (7.21)$$

$$Y = (y_{a1}, y_{a2}, \dots, y_{aM}) \quad (7.22)$$

where  $x_{Ri} \leq K, \forall i \in [1, N], y_{aj} \in V_B, a_j \in V_B, \forall j \in [1, M], K$  is an unknown number of layers to be inferred.

$X$  represents the partition of  $V_R$  into  $K$  layers with the partial occlusion relations represented in a Hasse diagram along with  $Y$ . The assignments among address variables indicate to whom the open bonds  $a_i$  are connected or assigned. Each terminator has the same label for its layer information as the region to which it is assigned.

Because the surface is merged from regions so that its generative model includes two constraints: one is appearance defined as the region model and the other shape constrained by some generic shape priors (Elastica in this chapter) or some specific object templates. Recovered contours are computed using Elastica based on the results of assignments.

### 7.3.2 The mixed MRF and Its Graphical Representation

Given the 2D representation defined above, there are two kinds of nodes in its graphical representation: region nodes (the nodes found in a traditional MRF) and terminator nodes (a newly introduced set of address nodes). Terminator nodes are dynamically broken from 3-degree-junctions in the process of region layering, which means that their neighborhood systems are determined on the fly. According to the discussion stated above, we know that this will build up a reconfigurable graph with a mixed MRF as illustrated in Figure 7.3 (b). We clarify some definitions in this section.

Let  $G = \langle V, E \rangle$  be the graph for the 2D representation, where  $V = V_R \cup V_B$  is the set of vertices and  $E = E_R \cup E_B$  the set of edges.  $E_R$  is the set of edges linking regions into a region adjacency graph and  $E_B$  is the dynamic set of edges linking open bonds.

The edges decide the topology of the graph, and can link, generally speaking, any two vertices. In a mixed MRF, the introduction of address variables makes the neighborhood system dynamic and different from traditional graphical models.

A mixed MRF can be defined in the same way that an MRF is defined as a probability distribution that is factorized into a product of local terms, the only difference being in what “local” means after introducing address variables. The idea of “locality” in a mixed MRF can handle long-range interactions through open bonds, meaning that a clique denoted by  $C$  in a mixed MRF may contain both standard region nodes and address nodes.

**Definition 1. Mixed neighbor potential.** Let  $C$  denote the set of cliques in  $G$ . A family of nonnegative functions  $\lambda_C : C \in \mathbb{C}$  is called a mixed neighbor potential if for any pair of configurations  $x$  and  $y$ ,  $x_C = y_C$  and  $x_{a_a} = y_{y_a}, \forall a \in C \cap V_B$ . Thus a mixed potential function  $\lambda_C$  depends on both the values of the standard nodes in  $C$  and that of those pointed to by open bonds in  $C$ .

**Definition 2. Mixed Markov random field.** A probability distribution  $P$  defined on  $G$  is a mixed MRF if  $P$  can be factorized into a product of mixed potential functions over the cliques:  $P(I) \propto \prod_{C \in \mathbb{C}} \lambda_C(I_C, I_{I_C})$ , where  $I_{I_C}$  is the vector of states of those standard variables pointed to from within the clique  $C$ .

**Equivalence between the mixed MRF and the Gibbs distribution.** It was shown that the originally established equivalence between MRFs and the Gibbs distribution by Hammersley-Clifford theorem, is also applicable for mixed MRFs so that the probability models defined on a mixed MRF can be simulated by Gibbs sampling [57] or SW-cuts [8] according to A.Fridman’s proof in [57].

Given a region  $R_i$ , its neighborhood is:

$$N(R_i) = N_{\text{adjacency}}(R_i) \cup N_{\text{pointer}}(R_i) \quad (7.23)$$

where  $N_{\text{adjacency}}(R_i) = \{R_j : \forall R_j \in V_R \text{ is adjacent to } R_i\}$  represents the usual definition of a neighborhood found in MRF models and defines an adjacency graph. At the same time, open bonds of  $R_i$  will cause it to link to those regions which are not locally adjacent, that is,  $N_{\text{pointer}}(R_i) = \{R_j : B(R_j) \text{ and } B(R_i) \text{ are connected}\}$ . Then  $E_R$  is:

$$E_R = \{\langle R_i, R_j \rangle, \forall R_i \in V_R \text{ and } R_j \in N(R_i)\} \quad (7.24)$$

Given an open bond  $a_i \in V_B$ , its neighborhood is:

$$N(a_i) = \{a_j : a_j \text{ and } a_i \text{ are in the same layer}\} \quad (7.25)$$

Hence  $E_B = \langle a_i, a_j \rangle, \forall a_i \in V_T \text{ and } a_j \in N(a_i)$ . At the initial stage, all the open bonds are open.



### 7.3.3 Bayesian Formulation

In the Bayesian framework, the three-layer model in the Eqn. 7.1 is described as

$$p(I, W_{2D}, W_{2.1D}) = p(I|W_{2D}, W_{2.1D})p(W_{2D}|W_{2.1D})p(W_{2.1D}) \quad (7.26)$$

where  $p(I|W_{2D}, W_{2.1D})$  is the likelihood model. We want to maximize the posterior joint probability of  $W_{2.1D}$  given  $W_{2D}$  in a solution space  $\Omega_{W_{2.1D}}$

$$\begin{aligned} W_{2.1D}^* &= \arg \max_{\Omega_{W_{2.1D}}} p(W_{2.1D}|W_{2D}; I) \\ &= \arg \max_{\Omega_{W_{2.1D}}} p(W_{2D}|W_{2.1D})p(W_{2.1D}) \end{aligned} \quad (7.27)$$

**A Graph Partition Perspective.** Given the graphical representation  $G$  defined with a mixed MRF, we are interested in a partitioning, or coloring, of the vertex, *i.e.*,  $V = V_R \cup V_B$ , in  $G$ . An  $n$ -partition is denoted by

$$\pi_n = (V_1, V_2, \dots, V_n), \bigcup_{i=1}^n V_i = V, V_i \cap V_j = \emptyset, \forall i \neq j \quad (7.28)$$

Each subset  $V_i, i = 1, 2, \dots, n$  (that is a surface in 2.1D representation) is assigned a color  $c_i$  which represents its model. For region nodes, this model consists of layering information and for open bonds the model consists of the connected contours. Let  $\Omega_{\pi_n} (\pi_n \in \Omega_{\pi_n})$  be the space of all possible  $n$ -partitions of  $V$ ,  $\Omega_{l_r}$  the set of types of region models,  $\Omega_{\theta_r}$  the model parameter family,  $\Omega_{l_c}$  the set of types of Elastica, and  $\Omega_{\theta_c}$  the Elastica parameter space. Thus, the solution space for  $W_{2.1D}$  is

$$\Omega = \bigcup_{n=1}^N \{ \Omega_{\pi_n} \times \Omega_{l_r}^n \times \Omega_{\theta_r} \times \dots \times \Omega_{\theta_{rn}} \times \Omega_{l_c}^n \times \Omega_{\theta_{c1}} \times \dots \times \Omega_{\theta_{cn}} \} \quad (7.29)$$

This leads us to extend the Swendsen-Wang cuts algorithm [8] to perform the inference. In the formulation, the prior model is

$$p(W_{2.1D}) = p(K)p(X)p(Y) \prod_{i=1}^N p(R_i) \quad (7.30)$$

where  $p(K)$  is an exponential model  $p(K) \propto \exp(-\lambda_0 K)$ ,  $p(X) = \prod_{i=1}^N p(X_{R_i})$  and  $p(Y) \propto \exp -\beta_0 \sum_{\forall a_i, a_j, i \neq j} 1(y_{a_i} = y_{a_j})$  penalize the situation that more than one terminator is assigned to the same junction and  $p(R_i)$  defined in Eqn. 12.68.

The likelihood model is

$$\begin{aligned} p(W_{2D}|W_{2.1D}) &\propto \prod_{i=1}^K \prod_{j=1}^{K(i)} \prod_{(x,y) \in s_j} H(I(x, y) - J(x, y; \theta_j)) \\ &\quad \times \prod_{i=1}^M \exp\{-\lambda_0 E\} \end{aligned} \quad (7.31)$$

where  $J(x, y; \theta_j)$  is defined in Eqn. 7.16 and  $\lambda_0$  is a scalable factor and also learned from the data set [253]. The first term in the likelihood model handles the region coloring/layering problem, and the second term handles the contour completion problem. They are solved jointly.

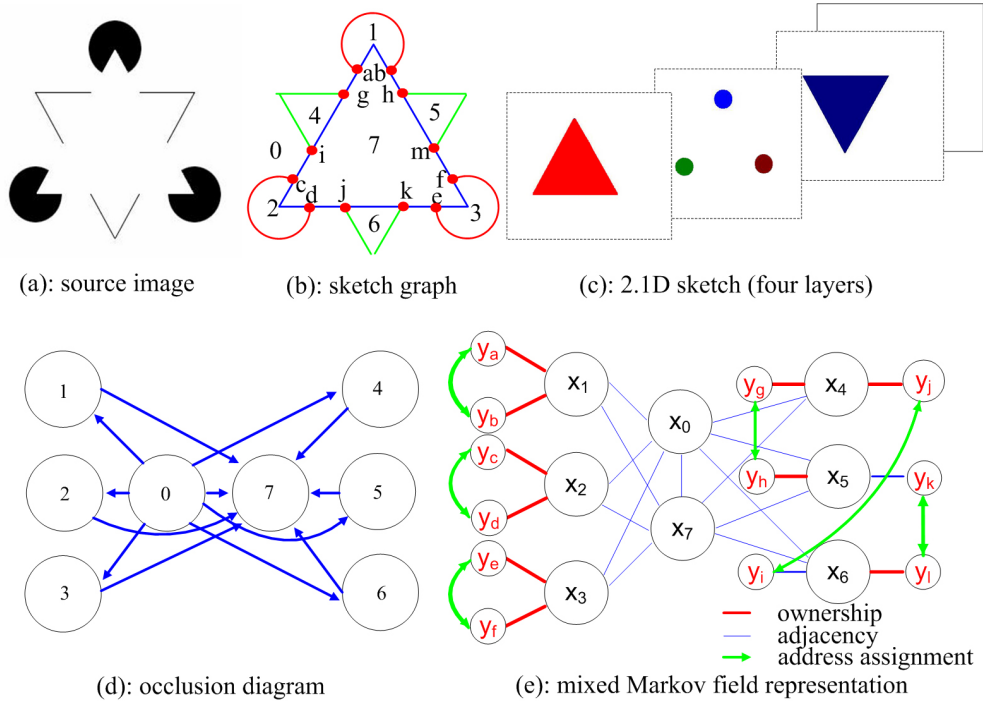


Figure 7.5: Reprinted with permission from [64]. A running example of inferring the 2.1D sketch. (a) is the original image, Kanizsa figure. (b) is the sketch graph computed by the primal sketch model after interactively labeling. There are 8 atomic regions and 12 terminators broken form T-junctions. (c) is the 2.1D sketch. There are 4 layers and the contour completion is performed using Elastica rules. (d) is the Hasse diagram for partial order relation, such as  $\langle 7, 1 \rangle$  means region 7 occludes region 1. (e) is the graph representation with a mixed Markov random field. Each big circle denotes a vertex of the atomic region, each red bar denotes one terminator, each little circle denotes a vertex of open bound described as address variable. Each region may have two or more terminators. The blue line segments connect the two neighboring regions. The green two-way arrows connect two terminators and each terminator is assigned another terminator's address variable.

**The Inference Algorithm.** Based on the graphical representation with a mixed MRF, the presented inference algorithm proceeds in two ways (1) region coloring/layering based on the Swendsen-Wang Cuts algorithm [8] for the partitioning of the region adjacency graph to obtain partial occluding order relations; (2) address variable assignments based on Gibbs sampling for the completion of open bonds. The basic goal is to realize a reversible jump between any two successive states  $\pi$  and  $\pi_0$  in terms of the Metropolis-Hastings method.

**Experiments.** We first demonstrate the inference algorithm using the Kanizsa image. Figure 7.5 shows the 2D representation and the graphical representation with mixed MRF. Figure 7.6 shows the inference procedure.

## 7.4 2.1D Sketch with Layered Regions and Curves

In this section, we handle images with both regions and curves which are frequently observed in natural images as illustrated in Figure 7.7. Given an input image, our objective is to infer an unknown number

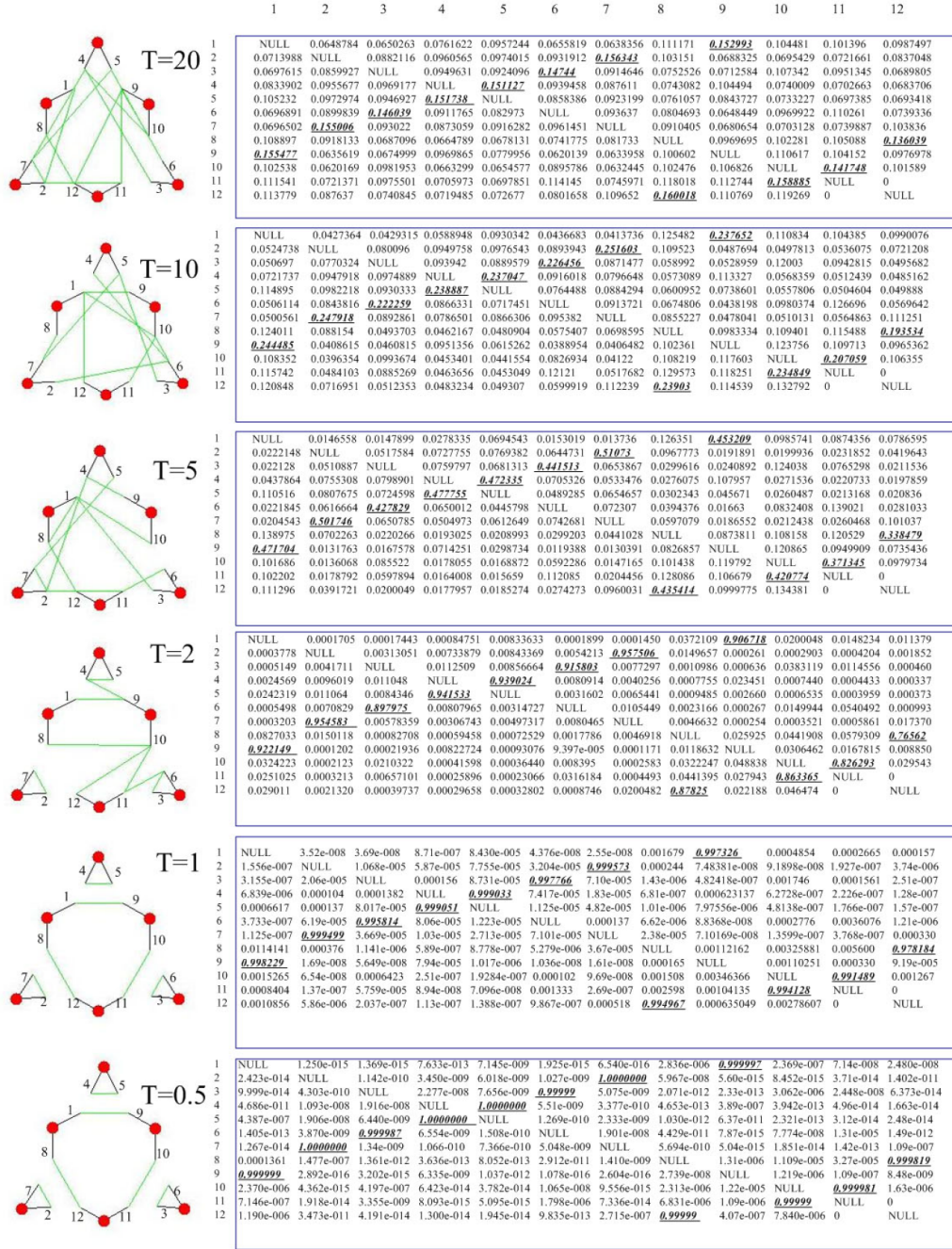


Figure 7.6: Reprinted with permission from [64]. Illustration of the Gibbs inference procedure. Each red node denotes a region, the black line segment denotes terminator, and the green line segment shows the inferred connection or assignment of address variables. Inference starts from a initial temperature  $T = 20$ ,  $(a) \sim (h)$  are the results in different temperatures. After  $T = 1$ , the result is right as in the figure.

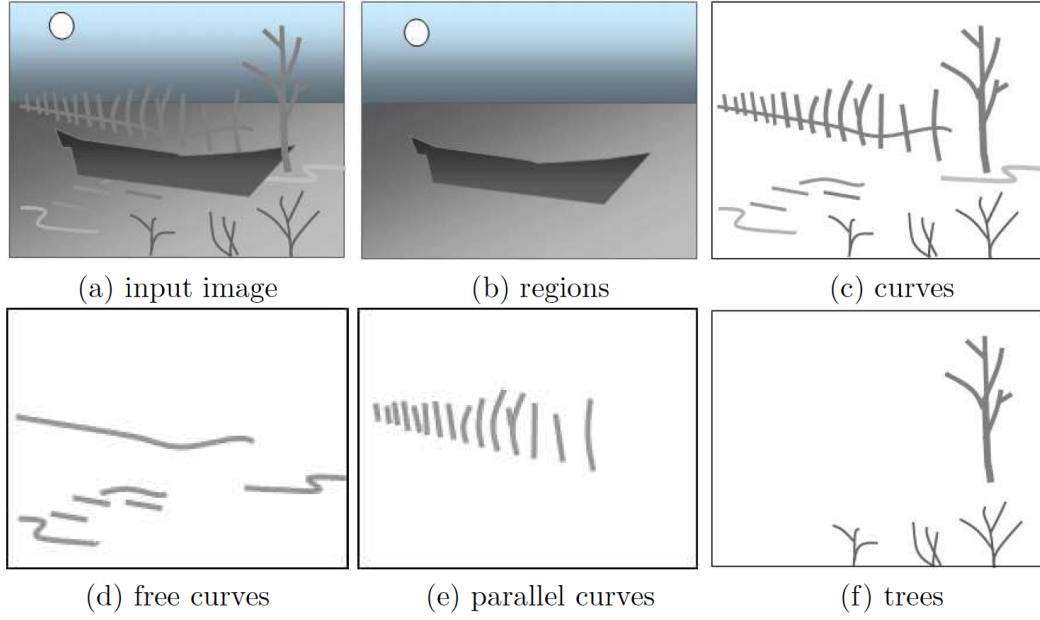


Figure 7.7: Reprinted with permission from [215]. Illustration of the 2.1D sketch with layered representation of regions and curves. (a) is an input image which is decomposed into two layers – (b) a layer of regions and (c) a layer of curves. These curves are further divided into (d) free curves, (e) a parallel curve group for the fence, and (f) trees. Curves observe a partial order occlusion relation.

of regions, free curves, parallel groups, and trees, with recovered occlusion relation and their probability models selected and fitted—all in the process of maximizing (or simulating) a Bayesian posterior probability. This algorithm searches for optimal solutions in a complex state space that contains a large number of subspaces of varying dimensions for the possible combinations of regions, curves, and curve groups.

### 7.4.1 Generative Models and Bayesian Formulation

In this section, we present generative models for both regions and curve structures.

#### Generative models of curves

We consider three types of curve models which are illustrated in Figure 11.12 and described as follows.

**1. Free curves.** A free curve, denoted by  $C$ , is represented by its medial axis  $\mathbf{c}_m(s) = (x_m(s), y_m(s))$  and its width  $2w(s)$  for  $s = [0, L]$ .  $L$  is the curve length. In a continuous representation, a free curve  $C$  occupies an elongated area or domain  $\mathcal{D}(C)$  bounded by the left and right side boundaries denoted respectively by  $\mathbf{c}_l(s) = (x_l(s), y_l(s))$  and  $\mathbf{c}_r = (x_r(s), y_r(s))$ . Fig. 11.12 (a) shows the boundaries in dashed lines.

$$\mathbf{c}_l(s) = \mathbf{c}_m(s) - w(s)\mathbf{n}(s), \mathbf{c}_r(s) = \mathbf{c}_m(s) + w(s)\mathbf{n}(s), \quad (7.32)$$

where  $\mathbf{n}(s)$  is the unit normal of  $\mathbf{c}_m(s)$ . Intuitively, a curve is a degenerated region parameterized by its 1D medial axis. Usually  $w(s)$  is only 1 – 3 pixels wide and  $w \ll L$ . This causes major topology problems in image segmentation where the two boundaries  $\mathbf{c}_l(s)$  and  $\mathbf{c}_r(s)$  could often intersect generating numerous trivial regions. This problem will be resolved with the explicit 1D representation. The intensities of a curve

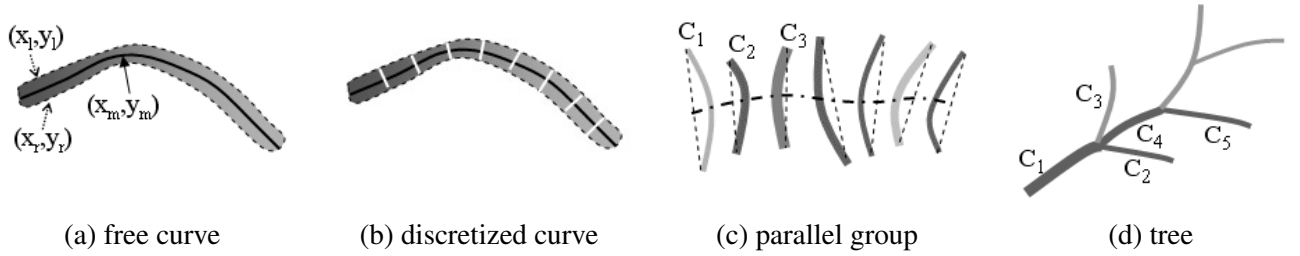


Figure 7.8: Reprinted with permission from [215]. Representations of curves and curve groups. (a) A free curve in continuous representation. (b) A free curve is discretized into a chain of “bars”. (c) Curves for a parallel group. (d) Curves for a Markov tree.

often exhibit globally smooth shading patterns, for example the curves in Figure 11.12. Thus we adopt a quadratic function for curve intensities,

$$\mathbf{J}(x, y; \theta_0) = ax^2 + bxy + cy^2 + dx + ey + f, (x, y) \in \mathcal{D}(C), \quad (7.33)$$

with parameters  $\theta_0 = (a, b, c, d, e, f)$ . The validation of choosing an inhomogeneous model to capture the smoothly changing intensity patterns can be found in [214]. Therefore, a free curve is described by the following variables in continuous representation

$$C = (L, \mathbf{c}_m(s)_{s=0}^L, w(s)_{s=0}^L, \theta_0, \sigma). \quad (7.34)$$

where  $\sigma$  is the variance of the intensity noise. While this continuous representation is a convenient model, we should also work on a discrete representation. Then the domain  $\mathcal{D}(C)$  is a set of pixels in a lattice and  $C$  is a chain of elongated bars as Figure 11.12 (b) illustrates.

The prior model for  $p(C)$  prefers smooth medial axes, narrow and uniform width, and it also has a term for the area of the curve in order to match with the region prior.

$$p(C) \propto p(\mathcal{D}(C))p(\mathbf{c}(s))p(w(s)) \propto e^{-E(C)}. \quad (7.35)$$

The energy  $E(C)$  is the sum of three terms

$$E(C) = \gamma_c |\mathcal{D}(C)|^\rho + \lambda L + E_o(w), \quad (7.36)$$

where  $\rho, \lambda$  are constants and are fixed in our experiments, and  $\gamma_c$  is a scale factor that can be adjusted to control the number of curves.  $E_o(w)$  is a term which constrains width  $w(s)$  to be small. We denote the intensities inside the curve domain by  $\mathbf{I}_{\mathcal{D}(C)}$ , and assume the reconstruction residue follows iid Gaussian  $\mathcal{N}(0; \sigma^2)$ . The image likelihood therefore is

$$p(\mathbf{I}_{\mathcal{D}(C)}|C) = \prod_{(x,y) \in \mathcal{D}(C)} \mathcal{N}(\mathbf{I}(x, y) - \mathbf{J}(x, y; \theta_0); \sigma^2). \quad (7.37)$$

**2. Parallel curve groups.** A parallel curve group consists of a number of nearly parallel curves as Fig. 11.12 (c) shows. Each curve  $C_i, i = 1, 2, \dots, n$  is summarized by a short line segment connecting its end points. They represent curve structures, such as zebra stripes, grids, and railings shown in the experiments. Grouping curves into a parallel group is encouraged in the model as it reduces the coding length and it is useful for perceiving an object, for example, a zebra. We denote a parallel curve group by

$$pg = (n, \{C_1, C_2, \dots, C_n\}, \{\alpha_1, \alpha_2, \dots, \alpha_n\}), \quad (7.38)$$

$\alpha_i \in \{1, \dots, n\}$  is the index to the curve preceding  $C_i$  in the chain.

The prior model for a  $pg$  is a first order Markov model in a Gibbs form with a singleton energy on individual curve and a pair energy for two consecutive curves as

$$p(pg) \propto \exp\{-\lambda_0 n - \sum_{i=1}^n E(C_i) - \sum_{i=2}^n E_{pg}(C_i, C_{\alpha_i})\}. \quad (7.39)$$

The singleton  $E(C_i)$  is inherited from the free curve model. For the pair energy, we summarize each curve  $C_i$  by five attributes: center  $(x_i, y_i)$ , orientation  $\theta_i$  of its associate line-segment, and length  $L_i$  of the line segment, curve average width (thickness)  $\bar{w}_i$ , and average intensity  $\mu_i$ .  $E_{pg}(C_i, C_{\alpha_i})$  measures the differences between these attributes.

**3. Markov trees.** Figure 11.12 (d) shows a number of curves in a Markov tree structure. We denote it by

$$T = (n, \{C_1, C_2, \dots, C_n\}, \{\beta_1, \beta_2, \dots, \beta_n\}). \quad (7.40)$$

$\beta_i \in \{1, \dots, n\}$  is the index to the parent curve of  $C_i$ . Thus the prior probability is

$$p(T) \propto \exp\{-\lambda_0 n - \sum_{i=1}^n E(C_i) - \sum_{\alpha_i \neq \emptyset} E_T(C_i, C_{\beta_i})\}. \quad (7.41)$$

Again,  $E(C_i)$  is inherited from the free curve. The term for  $C_i$  and its parent  $C_{\alpha_i}$ ,  $E_T(C_i, C_{\alpha_i})$ , measures the compatibility such as end-point gap, orientation continuity, thickness, and intensity between the parent and child curves.

The parallel group  $pg$  and tree  $T$  inherit the areas from the free curve, thus

$$\mathcal{D}(pg) = \cup_{i=1}^n \mathcal{D}(C_i), \text{ and } \mathcal{D}(T) = \cup_{i=1}^n \mathcal{D}(C_i). \quad (7.42)$$

It also inherits the intensity function  $\mathbf{J}(x, y; \theta_i)$  from each free curve  $C_i, i = 1, 2, \dots, n$ . In summary, the intensity models for  $C, pg, T$  are all generative for image  $\mathbf{I}$  as

$$\mathbf{I}(x, y) = \mathbf{J}(x, y; \theta) + \mathcal{N}(0; \sigma^2), (x, y) \in \mathcal{D}(C_i), \mathcal{D}(pg), \text{ or } \mathcal{D}(T). \quad (7.43)$$

### Generative models of regions

Once the curves explain away the elongated patterns, what is left within each image are the regions in the background. We adopt two simple region models in comparison to the four models in [214]. We denote a 2D region by  $R \subset \Lambda$  and  $\mathbf{I}_R$  the intensities inside  $R$ .

The first model assumes constant intensity with additive noise modeled by a non-parametric histogram  $\mathcal{H}$ .

$$\mathbf{J}(x, y; 1, \theta) = \mu, \mathbf{I}(x, y) = \mathbf{J}(x, y) + \eta, \eta \sim \mathcal{H}, (x, y) \in R. \quad (7.44)$$

With a slight abuse of notation, we denote by  $\theta = (\mu, \mathcal{H})$  the parameters used in a region.

The second model assumes a 2D Bezier spline function with additive noise. The spline accounts for global smooth shadings.

$$\mathbf{J}(x, y; 2, \theta) = B'(x)MB(y), \mathbf{I}(x, y) = \mathbf{J}(x, y; \theta_2) + \eta, \eta \sim \mathcal{H}, (x, y) \in R. \quad (7.45)$$

where  $B(x) = ((1-x)^3, 3x(1-x)^2, 3x^2(1-x), x^3)$  is the basis and  $M$  is a  $4 \times 4$  control matrix. This is to impose an inhomogeneous model for capturing the gradually changing intensity patterns, *e.g.*, the

sky. This model is important since regions with shading effects will be segmented into separate pieces with homogeneous models. The parameters are  $\theta = (M, \mathcal{H})$  and more details with a validation can be found in [214] where we compare different models for different types of images.

The likelihood probability is

$$p(\mathbf{I}_R|R, \theta) \propto \prod_{(x,y) \in \mathcal{D}(R)} \mathcal{H}(\mathbf{I}(x, y) - \mathbf{J}(x, y; \ell, \theta)), \quad \ell \in \{1, 2\}. \quad (7.46)$$

The prior for a region  $R$  assumes short boundary length  $\partial R$  (smoothness) and compact area  $|\mathcal{D}(R)|$ ,

$$p(R) \propto \exp\{-\gamma_r |\mathcal{D}(R)|^\rho - \frac{1}{2} \lambda |\partial R|\}, \quad (7.47)$$

where  $\rho$  and  $\lambda$  are constants that are fixed for all the experiments in this section, and  $\gamma_r$  is a scale factor that can be adjusted to control the number of regions in the segmentation.

## 7.4.2 Bayesian Formulation for Probabilistic Inference

Given an image  $\mathbf{I}$ , our objective is to compute a representation of the scene (world  $W$ ) in terms of a number of regions  $W^r$ , free curves  $W^c$ , parallel curve groups  $W^{pg}$ , trees  $W^t$ , and a partial order  $\mathcal{PR}$ . We denote the representation by variables

$$W = (W^r, W^c, W^{pg}, W^t, \mathcal{PR}). \quad (7.48)$$

The region representation  $W^r$  includes the number of regions  $K^r$ , and each region  $R_i$  has a label  $\ell_i \in \{1, 2\}$  and parameter  $\theta_i$  for its intensity model

$$W^r = (K^r, \{(R_i, \ell_i, \theta_i) : i = 1, 2, \dots, K^r\}). \quad (7.49)$$

Similarly, we have  $W^c = (K^c, C_1, \dots, C_{K^c})$ ,  $W^{pg} = (K^{pg}, pg_1, pg_2, \dots, pg_{K^{pg}})$ , and  $W^t = (K^t, T_1, T_2, \dots, T_{K^t})$ . In this model, there is no need to define the background since each pixel either belongs to a region or is explained by a curve/curve group.

The problem is posed as Bayesian inference in a solution space  $\Omega$ .

$$W^* = \arg \max_{W \in \Omega} p(\mathbf{I}|W)p(W). \quad (7.50)$$

By assuming mutual independence between  $W^r, W^c, W^{pg}, W^t$  we have the prior model

$$p(W) = \left(p(K^r) \prod_{i=1}^{K^r} p(R_i)\right) \left(p(K^c) \prod_{i=1}^{K^c} p(C_i)\right) \left(p(K^{pg}) \prod_{i=1}^{K^{pg}} p(pg_i)\right) \left(p(K^t) \prod_{i=1}^{K^t} p(T_i)\right). \quad (7.51)$$

The prior for individual  $p(R), p(C), p(pg), p(T)$  are given in the previous subsections.

As there are  $N$  curves in total including the free curves, and curves in the parallel groups and trees, then the likelihood model follows the lattice partition and Eqn. 7.37 and 7.46.

$$p(\mathbf{I}|W) = \prod_{i=1}^{K^r} \prod_{(x,y) \in \Lambda_{R_i}} \mathcal{H}((\mathbf{I}(x, y) - \mathbf{J}(x, y; \ell_i, \theta_i))) \cdot \prod_{j=1}^N \prod_{(x,y) \in \Lambda_{C_j}} \mathcal{N}((\mathbf{I}(x, y) - \mathbf{J}(x, y; \theta_j); \sigma_j^2)). \quad (7.52)$$

Since all objects use generative models for reconstructing  $\mathbf{I}$ , these models are directly comparable and they compete to explain the image. This property is crucial for the integration of region segmentation and curve grouping.



**The Inference Algorithm.** Here we briefly summarize the design of the algorithm. The goal is to design an algorithm to make inference of the  $W^*$  which maximizes the posterior  $p(W|\mathbf{I})$  by sampling  $W$  in the solution space with a fast simulated annealing procedure. Since  $W^*$  is usually highly peaked, we hope that it will most likely be sampled if the algorithm converges to the target distribution. This poses rather serious challenges even though we have simplified the image models above. The main difficulty is to deal with objects with different structures and explore a large number of possible combinations of regions, curves, and curve groups in an image. Especially our objective is to achieve automatic and nearly globally optimal solutions. The following are three basic considerations in our MCMC design.

Firstly, the Markov chain should be irreducible so that it can traverse the entire solution space. This is done by designing a number of pairs of jumps to form an ergodic Markov chain. The resulting Markov chain can reach any state from an arbitrary initialization.

Secondly, each jump operates on 1–2 curves or curve elements. We study the scopes of the jumps within which the algorithm proposes the next state according to a conditional probability. This is like a Gibbs sampler. The proposal is then accepted in a Metropolis-Hastings step, hence its name, the *Metropolized Gibbs Sampler* (MGS [141]).

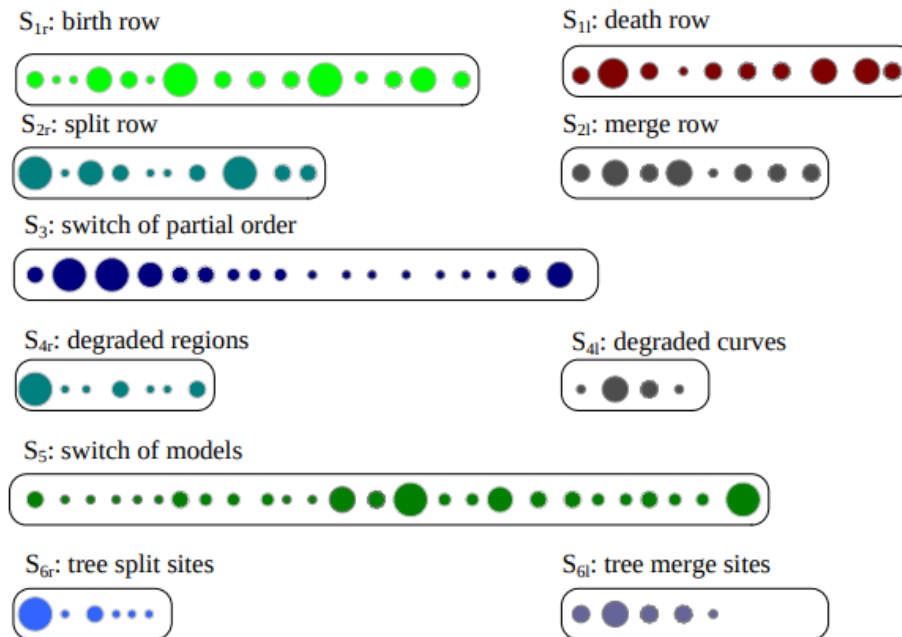


Figure 7.9: Reprinted with permission from [215]. The 6 simple jumps maintain 10 sets of “particles” whose sizes illustrate their weights. The sets are updated and re-weighted in each jump steps, and they encode the proposal probabilities in a non-parametric representation.

Thirdly, the computational cost at each jump step should be small. The proposal probability *ratios* in our design are factorized and computed by discriminative probability ratios. These discriminative probabilities are computed in bottom-up processes which are then used to activate the generative models in a top-down process. As Fig. 7.9 illustrates, each jump maintains a list of “particles” which are weighted hypotheses with the weights expressing the discriminative probability ratios. Then a particle is proposed at a probability proportional to its weight within the list (scope). The higher the weight is, the more likely a particle will be chosen.



### 7.4.3 Experiments

The proposed algorithm searches for the optimal solution  $W^*$  by sampling  $p(W|I)$ . It starts from a segmentation with regions obtained at a coarse level by the Canny edge detector. Our method does not rely much on the initial solution due to the use of various MCMC dynamics guided by bottom-up proposals, which help the algorithm to jump out of local minimums. However, we do use an annealing strategy to allow large change of  $W$  at high temperatures, and to focus more on local modes with the temperature gradually cooling down. The optimal solution  $W^*$  is found when the algorithm converges since  $p(W^*|I)$  is in general highly peaked for many vision problems, especially at a low temperature. It is always desirable to avoid the use of annealing.

#### Experiment A: computing regions and free curves

An example is shown in Fig. 7.10. For each example, the first row displays the input image  $\mathbf{I}_{\text{obs}}$ , the computed free curves  $W^c$ , and the region segmentations  $W^r$  in the background. The second row shows the synthesized image according to the generative models for the regions  $\mathbf{I}_{\text{syn}}^r \sim p(\mathbf{I}|\mathbf{W}^r)$ , the curves  $\mathbf{I}_{\text{syn}}^c \sim p(\mathbf{I}|W^c)$ , and the overall synthesis  $\mathbf{I}_{\text{syn}}$  by occluding  $\mathbf{I}_{\text{syn}}^c$  on  $\mathbf{I}_{\text{syn}}^r$ .

We construct a synthesis image to verify how an input image is represented in  $W^*$ . In the experiment, two parameters in the prior models are adjustable: (1)  $\gamma_r$  in Eqn. 7.47, and (2)  $\gamma_c$  in Eqn. 7.36. The two parameters control the extent of the segmentation, *i.e.*, the number of regions and curves. Therefore they decide how detailed we like the parsing to be. Usually we set  $\gamma_r = 5.0$  and  $\gamma_c = 3.5$  and other parameters are fixed.

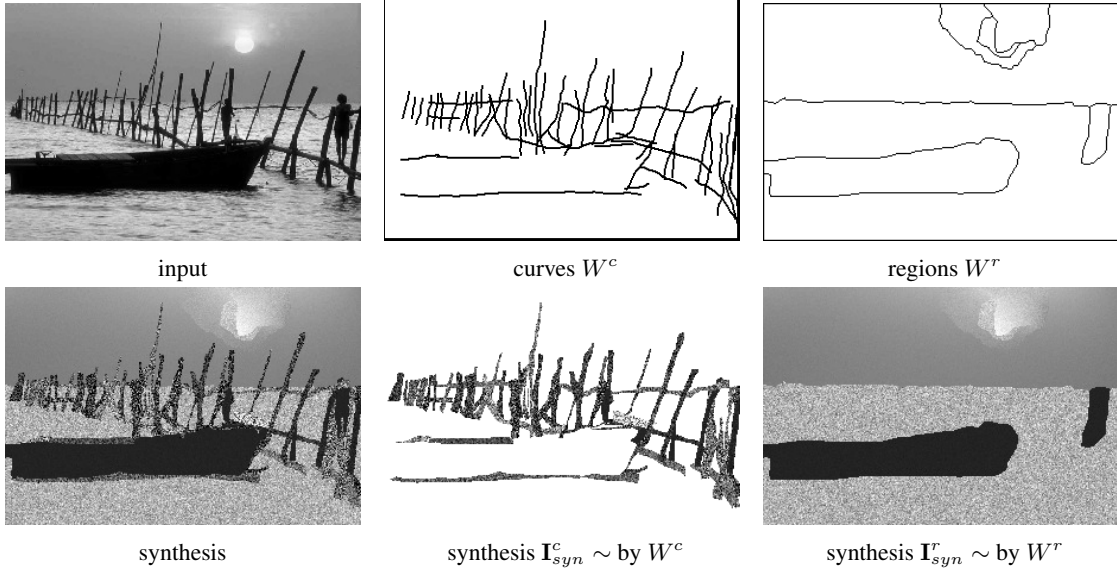


Figure 7.10: Reprinted with permission from [215]. Experiment A: parsing images into regions and free curves.

In the second experiment, we further compute the parallel groups and trees by turning on the two composite jumps  $\mathbf{J}_7, \mathbf{J}_8$ . Figures 7.11 shows an example: The top row shows the parallel groups or trees grouped from the simple curves. The second and third rows are displayed as before. From the results, we can see that the algorithm successfully segments, detects, and groups regions, curves, and curve groups respectively.

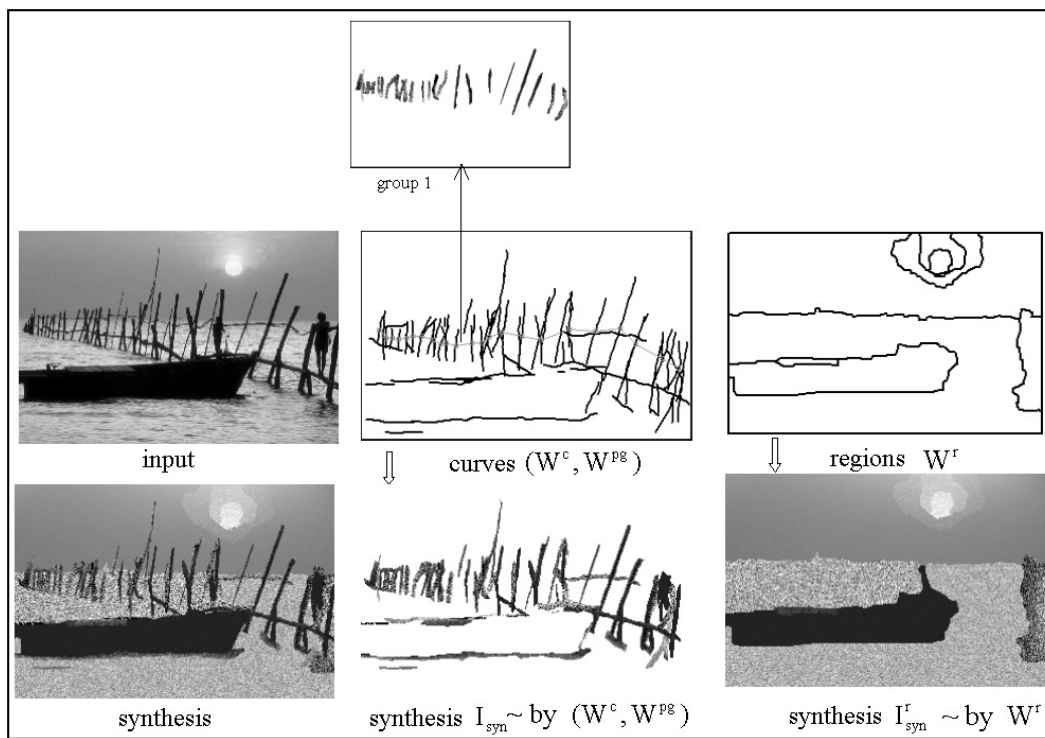


Figure 7.11: Reprinted with permission from [215]. Experiment B: parsing an image into regions, curves, and parallel curve groups.

## 8

# 2.5D Sketch and Depth Maps

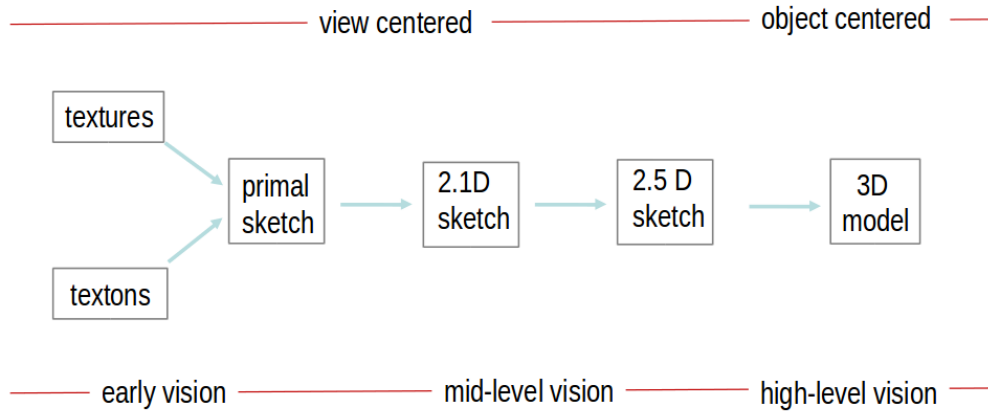


Figure 8.1: Representation in Marr’s paradigm. The 2.1D sketch was proposed by Nitzberg and Mumford [169]

In previous chapters, we discuss the first stage of early visual processing, i.e., representing the changes and structures in the image with the primal sketch and 2.1D sketch. In general, the primal sketch is a generic two-layer 2D representation describing image features such as intensity changes, local geometrical structures, and illumination effects like light sources, highlights, and transparency. Based on the primal sketch, the 2.1D sketch, a layered representation, is analyzed to describe the surfaces with occluding relations, defining the visibility of surfaces and contours in the given image. However, such rough descriptions of the spatial relations in images are not sufficient for our overall goal to thoroughly understand the vision. To understand how to obtain descriptions of the world efficiently and reliably from images, we need to make one step further. In this chapter, we introduce the 2.5D sketch, which aims to analyze the depth information that describes the relative positions of surfaces in a more precise way.

The idea of the 2.5D sketch first appeared in Marr and Nishihara’s research [154], whose original goal is to provide a viewer-centered representation of the visible surfaces. As shown in Figure 8.1, the 2.5D sketch construction is considered to be a significant part of the mid-level vision, the last step before surface interpretation, and the end, perhaps, of pure perception. We start from Marr’s definition of 2.5D sketch and the viewer-centered representation, to analyze the vision process from the primal sketch to 2.5D sketch in a relatively intuitive perspective. Besides, we introduce some methods to reconstruct the 2.5D presentation,

i.e., the depth information, for image analyzing, including the *shape from stereo*, *shape from shading* and *direct estimation*.

## 8.1 Marr's Definition

As introduced by Marr in [153], the distances from the observed objects to human's two eyes are different, making the two eyes form different images of the world. The disparities in the two images help human brains estimate the relative distances of the objects from the observer. Marr divides this account into two parts, the first concerned with measuring disparity and the second concerned with using it. Marr separates both parts into three levels, i.e., computational theory, representation and algorithm, and hardware implementation. More specifically, the three steps involved in measuring stereo disparities are: 1) selecting a particular location on a surface in the scene from one image; 2) identifying the same location in the other image; and 3) measuring the disparity between the two corresponding image points. Before making use of the measured disparity, we need to solve another fundamental problem in binocular fusion, i.e., the elimination or avoidance of false targets. The abundance of matchable features and the disparities between these matches lead to difficulties in deciding which pair is false and useless. With the validate measured disparities, we can compute the distance from the viewer to surfaces in the images, and also analyze the surface orientations from the disparity changes.

According to Marr, constructing the orientation-and-depth map of the visible surfaces around a viewer is one of the most critical goals of early visual processing. In this map, information is combined from several different and probably independent processes to interpret disparity, motion, shading, texture, and contour information. These ideas are called 2.5D sketch by Marr and Nishihara [154]. The full 2.5D sketch would include rough distances to the surfaces as well as their orientations. Besides, the places where surface orientations change sharply and where depth is discontinuous could be labeled as contours, which are informative for further analysis.

Many types of information can be extracted from images by different kinds of early visual processes. For example, the stereopsis outputs disparity, hence the continuous or small local changes in relative depth. At the same time, the optical flow generates the relative depth and the local surface orientation. Although in principle processes like stereopsis and motion can directly deliver depth-related information, they are more likely to deliver local changes of depth by measuring local disparities in practice. However, the main function of the visual representation we seek should not be limited to making explicit information about depth, local surface orientations, and discontinuities in these quantities. We want to create and maintain a global representation of depth that is consistent with the local cues obtained from these sources. Such a representation is called the 2.5D sketch.

To prepare for a more thorough discussion, we first describe the original definition for a viewer-centered representation that uses surface primitives of small size as described in [153]. Figure 8.2 illustrates such representation, which is like having a gradient space at each point in the visual field. It includes a representation of contours of surface discontinuity. It has enough internal computational structure to maintain its descriptions of depth, surface orientation, and surface discontinuity in a consistent state. Depth may be represented by a scalar quantity  $r$ , the distance from the viewer of a point on a surface, and the surface discontinuities may be represented by oriented line elements. Surface orientation may be represented as a vector  $(p, q)$  in two-dimensional space, equivalent to covering the image with needles. The length of each needle defines the surface slant (or dip) at the point. Here zero-length corresponds to a surface that is perpendicular to the vector from the viewer to that point. The length of the needle increases as the surface slants away from the viewer. The needle orientation defines the tilt, that is, the direction of the surface's slant. In principle, the relationship between depth and surface orientation is straightforward—one is simply

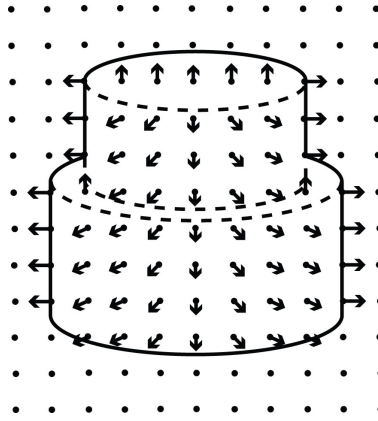


Figure 8.2: Reprinted with permission from [154]. Example of a 2.5D Sketch described in [153]. The surface orientation is represented by arrows, as explained in the text. Occluding contours are shown with full lines, and surface orientation discontinuities with dotted lines. Depth is not shown in the figure, though it is thought that rough depth is available in the representation.

the integral of the other, taking over regions bounded by surface discontinuities. Therefore, it is possible to devise a representation with intrinsic computational facilities that can maintain the two variables of depth and surface orientation in a consistent state.

We will discuss the approaches to generate the 2.5D sketch in the following sections, including two generative methods (in applications of shape from stereo and shape from shading) that generate the 2.5D sketch from the primal sketch and a recent discriminative approach in deep learning.

## 8.2 2.5D Sketch from Primal Sketch — Shape from Stereo

In this section, we introduce two works related to shape from stereo. We first briefly introduce the stereopsis work proposed by Peter Belhumeur in [12]. After that, we introduce the visual knowledge representation applied in the stereo reconstruction work proposed by [3].

A pair of stereo images have been known to encode the information detailing the scene geometry since at least the time of Leonardo da Vinci in 1498 [30]. The animal brain has known this for millions of years and has developed yet barely understood neuronal mechanisms for decoding it. Everywhere animals gaze, they are able to aware of the relative depths of the observed objects. Even though the stereo vision is not the only cue to depth, monocular cues are still less exact and often ambiguous in determining the depth and 3D spatial relations.

Binocular stereopsis algorithms use the data in a pair of images taken from slightly different viewpoints to construct a depth map of the 3-D surfaces captured within the images [12]. The 3-D surfaces are estimated by first matching pixels in the images that correspond to the same point on a 3-D surface, and then computing the point's depth as a function of its displacement (or disparity) in the two images. The task of matching points between the two images is known as the correspondence problem.

In the last fifty years, researchers have tried to reconstruct the scene geometry from a pair of stereo images. Unfortunately, like most computer vision problems, the stereo problem has proven to be more difficult than initially anticipated. Argued by [12], the solutions for properly handling occluded regions and salient features in the scene geometry have been largely overlooked. Generally speaking, there are the following four major problems in matching the two images in binocular stereopsis. 1) Handling the

feature noise in the left/right images caused by quantization error, imperfect optics, imaging system, lighting variation between images, specular reflection, etc. 2) Handling the indistinct image features near pixels where the matching is ambiguous. 3) Preserving the salient features, such as depth discontinuities at object boundaries, in the 3D scene geometry to produce an accurate reconstruction. 4) Correctly handling the half-occlusion regions by first matching half-occluded points to mutually visible points and then estimating the depth at these points.

For years people have offered solutions to the correspondence problem without adequately addressing all of these complications. In the early area-based algorithms, the disparity was assumed to be constant within a fixed-sized window. Other methods integrate a type of smoothness (flatness) constraint into the matching process, again biasing toward reconstructions with persistent disparity. Although algorithms using smoothness constraints show effectiveness in handling the first two problems, their performance deteriorated at salient features in the scene geometry. These algorithms generally over-smooth the depth discontinuities at object boundaries ("breaks") and in surface orientation ("creases"). Sometimes erratic results would be produced. Line processes were introduced to solve the image segmentation problem. While this method helps preserve the boundaries of objects, it overlooks three critical complications. First, the introduction of line processes to model object boundaries gave rise to highly non-linear optimization problems. Second, no prescription was given for preserving the other salient features in the scene geometry. Third, it is challenging to identify whole regions of half-occlusion caused by discontinuity.

Facing the previous challenges, [12] developed a computational model for stereopsis by making explicit all of the assumptions about the nature of image coding and the structure of the world. In designing computer vision models, researchers often skip this step and, consequently, have no way of testing whether the underlying assumptions are valid. The developed model in [12] is designed within a Bayesian framework, and attempts to explain the process by which the information detailing the 3-D geometry of object surfaces is encoded in a pair of stereo images. [12] starts by deriving the model for image formation, introducing a definition of half-occluded regions and deriving simple equations relating these regions to the disparity function. In this work, the author shows that the disparity function alone contains enough information to determine the half-occluded regions. These relations are utilized to derive a model for image formation in which the half-occluded regions are explicitly represented and computed. The prior model is presented in a series of three stages, or "worlds," where each world considers an additional complication to the prior, constructed from all of the local quantities in the scene geometry, i.e., depth, surface orientation, object boundaries, and surface creases.

For computer vision problems, the Bayesian paradigm seeks to extract scene information from an image or sequence of images by balancing the content of the observed image with prior expectations about the observed scene's content. This method is general and can be applied to a wide range of vision problems, including binocular stereopsis. Let  $\mathbf{S}$  indicate the scene geometry given the left and the right images by  $\mathbf{I}_l$  and  $\mathbf{I}_r$ . Within the Bayesian paradigm, one infers  $\mathbf{S}$  by considering  $P(\mathbf{S}|\mathbf{I}_l, \mathbf{I}_r)$ , the posterior probability of the state of world given the measurement. From Baye's theorem, we have

$$P(\mathbf{S}|\mathbf{I}_l, \mathbf{I}_r) = \frac{P(\mathbf{I}_l, \mathbf{I}_r|\mathbf{S})P(\mathbf{S})}{P(\mathbf{I}_l, \mathbf{I}_r)} \quad (8.1)$$

Sometimes  $P(\mathbf{I}_l, \mathbf{I}_r|\mathbf{S})$  is referred to as the "image formation model" which measures how well  $\mathbf{S}$  matches the observed images.  $P(\mathbf{S})$  is usually referred to as the "prior model" that measures how probable a particular  $\mathbf{S}$  is a priori before the images are observed. We display the *maximum a posteriori*(MAP) estimate  $\hat{\mathbf{S}}$ . For notational convenience, we define the "energy" functional as below

$$E(\mathbf{S}) = -\log(P(\mathbf{I}_l, \mathbf{I}_r|\mathbf{S})P(\mathbf{S})) = E_D + E_P \quad (8.2)$$

where  $E_D = -\log(P(\mathbf{I}_l, \mathbf{I}_r|\mathbf{S}))$  is referred as the "data term", and  $E_P = -\log(P(\mathbf{S}))$  is referred as the "prior term". To employ these energy functionals with the underlying assumptions taken into consideration, one must be careful in choosing random variables to be estimated and in the assumed relations between these random variables in developing a Bayesian formulation of a vision problem.

### 8.2.1 The Image Formation Model

In this subsection, we introduce the image formation model proposed by [12]. Besides, the definition of the epipolar line will be mentioned.

In deriving the model for image formation, we choose the simplest possible geometry: pinhole cameras with parallel optical axes. Assume the cameras are calibrated and the epipolar geometry is known, we define disparity relative to an imaginary cyclopean image plane placed halfway between the left and right cameras. Here we derive explicit relations between disparity and depth, as well as disparity and half-occlusion, showing that the disparity function correctly determines the half-occluded regions in the left and right image planes. We then use these relations to derive our image formation model.

Assume that we have two pinhole cameras whose optical axes are parallel and separated by a distance of  $w$ . The cameras each have focal length  $l$ , with  $f_l$  the focal point of the left image, and  $f_r$  the right. Then we create an imaginary cyclopean camera in the same manner, placing its focal point  $f$  half-way along the baseline, i.e., the line connecting the left and right focal points. Then we restrict the cameras' placement so that the baseline is parallel to the image planes and perpendicular to the optical axes, as shown in 8.3. A point  $p$  on the surface of an object in 3D space which is visible to all three cameras is projected through the focal points onto the image planes. Each image plane has a 2D coordinate system with its origin determined by the intersection of its optical axis with the image plane. The brightness of each point projected onto the image planes creates image luminance functions  $\mathbf{I}_l$ ,  $\mathbf{I}_r$ , and  $\mathbf{I}$  in the left, right, and cyclopean planes, respectively.

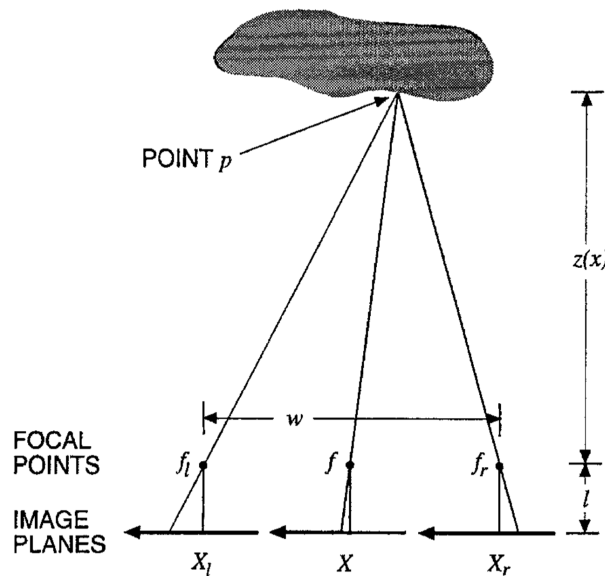


Figure 8.3: Reprinted with permission from [12]. Camera geometries: The figure shows the left and right image planes, plus an imaginary cyclopean image plane. Both the disparity and distance functions are defined relative to the cyclopean image plane.

A horizontal plane through the baseline intersects the three image planes in what are called *epipolar lines*, which we denote by  $X_l$ ,  $X_r$ , and  $X$ , with coordinates  $x_l \in X_l$ ,  $x_r \in X_r$ , and  $x \in X$ , respectively. The coordinates of the epipolar lines run from right to left, so that when a point in the world moves from left to right, its coordinates in the image planes increase. When the same point is visible from all three eyes, it is easy to check that  $x = (x_l + x_r)/2$ . Thus, we can relate the coordinates of points projected onto all three image planes by a positive disparity function  $d(x)$  via  $x_l = x + d(x)$  and a negative one via  $x_r = x - d(x)$ . Thus we have  $d(x) = \frac{x_l - x_r}{2}$ . Suppose  $z(x)$  represents the perpendicular distance from a line connecting the focal points to the point  $p$  on the surface of the object, then the disparity  $d(x)$  can be related to the distance  $z(x)$  by  $d(x) = \frac{lw}{2z(x)}$ .

Some early work assumed that none of the points in either of the left or right images are *half-occluded* (visible in one camera, but not in the other). However, the vast majority of the millions of images we view everyday contain large regions of half-occluded points, and half-occlusion is also a positive cue for human visual system to determine depth. Thus, computer vision systems must take advantage of the cues provided by half-occlusion as human visual system [12].

Here we give the definition of the mutually visible point as proposed in [12]. A point  $p$  is mutually visible to both eyes if the triangle formed by  $p$ ,  $f_l$  and  $f_r$  is free of obstructing objects, as shown in 8.4. Note that according to this definition, if any object is contained within the triangle formed by  $p$ ,  $f_l$  and  $f_r$ , then the point  $p$  is not considered mutually visible, even though the point may be visible to all three eyes. To determine from the disparity function when a point is mutually visible, it is convenient to introduce a morphologically filtered version  $d^*(x)$  of  $d(x)$ ,  $d^*(x) = \max(d(x + a) - |a|)$ .  $d^*(x) = d(x)$  if and only if the point  $p$  visible to the cyclopean eye in direction  $x$  is mutually visible to the left and right eyes. Thus, the function  $d^*(x)$  tracks the mutually visible points. Then we can define the half-occluded points as the points which are not mutually visible as in [12]. The half-occluded points  $O \in X$  are the closure of the set of points  $x$  such that  $d^*(x) > d(x)$ .

Before further derivations, we must define the previous quantities in the discrete manner. Take the fixed interval  $[-a, a]$  of the cyclopean epipolar line  $X$  and sample it at  $n$  evenly spaced points represented by  $X = x_1 \dots x_N$  such that  $x_1 = -a$ ,  $x_{i+1} - x_i = \delta$  and  $x_N = a$ . Let the disparities at the sampled points be represented by  $D = d(x_1) \dots d(x_N) = d_1 \dots d_N$ . We define the half-occluded points  $O \in X$  as  $O = \{x_i | d_j - d_i > |j - i| \text{ for some } x_j\}$ . Finally, we discretize the range of possible disparity value with sub-pixel fineness, so that  $d_i \in 0, \frac{1}{k}, \frac{2}{k}, \dots, 1, 1 + \frac{1}{k}, \dots, d_{max}$  for some  $k$  specifying the disparity resolution. Since it is not possible to distinguish between jumps in the disparity along a sloping surface and jumps in disparity at the boundaries of objects unless using sub-pixel resolution, these methods will falsely assume that the jumps along sloping surfaces produce half-occluded points. Yet, all surfaces visible to both the left and right eyes have  $[d_{i+1} - d_i] < 1$  as pointed out in [12].

Keeping within the Bayesian framework, a probabilistic model need to be developed for the joint distribution  $P(\mathbf{I}_l, \mathbf{I}_r | \mathbf{S})$ . To do this, assume we are given a scene of objects in 3D space with Lambertian illumination (i.e., an object's brightness is independent of the viewing angle). Label points on the surfaces of objects by elements of a set  $\Psi$ . To each point  $p \in \Psi$ , there is a brightness  $\gamma(p)$ . Define  $\prod_l$  and  $\prod_r$  to be the maps that take points in the image planes to the point on the surface of the closest visible object, i.e.,  $\prod_l : X_l \rightarrow \Psi$  and  $\prod_r : X_r \rightarrow \Psi$ . The brightness of a visible point once projected into the image plane is corrupted by noise. Assuming additive Gaussian white noise, image functions can be written as  $\mathbf{I}_l(x_l) = \gamma(\prod_l(x_l)) + \eta_l(x_l)$  and  $\mathbf{I}_r(x_r) = \gamma(\prod_r(x_r)) + \eta_r(x_r)$  where  $\eta_l$  and  $\eta_r$  are independent identically distributed (i.i.d.) Gaussian noise processes having mean zero and variance  $\nu^2$ . For notational convenience, we only consider the image functions  $\mathbf{I}_l$  and  $\mathbf{I}_r$  along corresponding epipolar lines. The joint density for any



set of  $N$  samples, which we denote by  $x_{l1}, \dots, x_{lN}$ , from the left image function  $I_l$ , given  $\gamma$ , is

$$P(\mathbf{I}_l(x_{l1}), \dots, \mathbf{I}_l(x_{lN})|\gamma) = P(\mathbf{I}_l|\gamma) = \frac{1}{(2\pi\nu^2)^{N/2}} \prod_{i=1}^N e^{-\frac{\eta_l^2(x_{li})}{2\nu^2}} \quad (8.3)$$

where  $\eta_l(x_{li}) = I_l(x_{li}) - \gamma(\prod_l(x_{li}))$ . Likewise, we can get the joint density from the right image function  $\mathbf{I}_r$ . Using the fact that  $\eta_l$  and  $\eta_r$  are independent, we can write the combined joint density as

$$P(\mathbf{I}_l, \mathbf{I}_r|\gamma) = \frac{1}{(2\pi\nu^2)^N} \prod_{i=1}^N e^{-\frac{\eta_l^2(x_{li}) + \eta_r^2(x_{ri})}{2\nu^2}}. \quad (8.4)$$

Choose the  $N$  samples from the left and right epipolar lines which correspond to the evenly spaced points

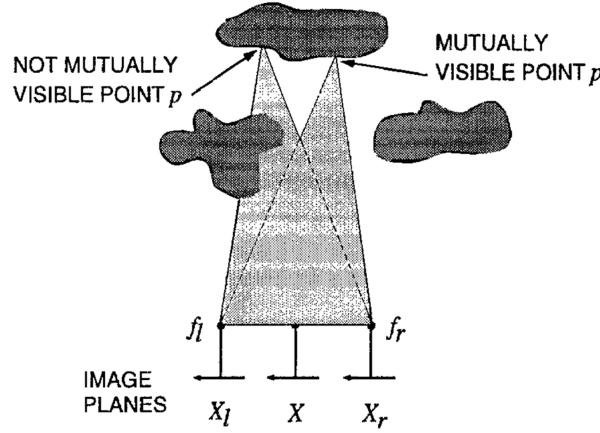


Figure 8.4: Reprinted with permission from [12]. Mutually visible points: A mutually visible point has no object within the triangle specified by  $p$ ,  $f_l$ , and  $f_r$ .

$x_1, \dots, x_N$  on the cyclopean epipolar line. So we choose  $x_{li} = x_i + d_i$  and  $x_{ri} = x_i - d_i$ . Because the brightness function  $\gamma$  is unknown, we approximate  $\gamma$ , with its maximum likelihood estimator  $\hat{\gamma}(\prod_l(x_{li})) = \hat{\gamma}(\prod_r(x_{ri})) = \frac{\mathbf{I}_l(x_{li}) + \mathbf{I}_r(x_{ri})}{2}$ . This approximation yields  $\eta_l^2(x_{li}) + \eta_r^2(x_{ri}) \approx \frac{(\mathbf{I}_l(x_{li}) - \mathbf{I}_r(x_{ri}))^2}{2}$ , then we will have the joint density  $P(\mathbf{I}_l, \mathbf{I}_r|\hat{\gamma}) = \frac{1}{(2\pi\nu^2)^N} \prod_{i=1}^N e^{-\frac{(\mathbf{I}_l(x_{li}) - \mathbf{I}_r(x_{ri}))^2}{4\nu^2}}$ . In this way, we can compute this quantity from the data if the point  $x_i$  is mutually visible. However, if the  $x_i$  is half-occluded, such formulation is not feasible.

To solve for the half-occluded points, we approximate the squared difference  $(\mathbf{I}_l(x_{li}) - \mathbf{I}_r(x_{ri}))^2/2$  by its expected value  $\nu^2$ . Then we have the combined joint density as below

$$P(\mathbf{I}_l, \mathbf{I}_r|\hat{\gamma}) = \frac{1}{(2\pi\nu^2)^N} \prod_{i=1, x_i \notin O}^N e^{-\frac{(\mathbf{I}_l(x_{li}) - \mathbf{I}_r(x_{ri}))^2}{4\nu^2}} \prod_{i=1, x_i \in O}^N e^{-\frac{1}{2}}. \quad (8.5)$$

or equivalently, the combined joint distribution can be rewrote in terms of the cyclopean epipolar points  $X$  and the corresponding disparities  $D$  as

$$P(\mathbf{I}_l, \mathbf{I}_r|\hat{\gamma}, D) = \frac{1}{(2\pi\nu^2)^N} e^{-E_D}, E_D = \frac{1}{4\nu^2} \sum_{X \in O} (\mathbf{I}_l(x_i + d_i) - \mathbf{I}_r(x_i - d_i))^2 + \sum_O \frac{1}{2} \quad (8.6)$$

with  $E_D$  as the data term in the model.

We have introduced the data model under the assumption of Lambertian illumination so far. Let us generalize the above equation so that the data term considers, as opposed to simply image intensity, other, possibly more viewpoint invariant, features (e.g., edges, texture, filtered intensity, etc.). In doing this we rewrite the above equation by replacing the intensity functions  $I_l$  and  $I_r$  with general feature functions  $F_l$  and  $F_r$ . Thus, the data term becomes  $E_D = \frac{1}{4\nu^2} \sum_{X-O} (F_l(x_i + d_i) - F_r(x_i - d_i))^2 + \sum_O \frac{1}{2}$ .

Furthermore, [12] derives the prior model for the Bayesian estimator, arguing that to capture the quantities in the scene geometry—namely depth, surface orientation, object boundaries, and surface creases—one should explicitly represent these quantities as random variables or continuous-time random processes in the prior model. The derivation is broken up into three stages, or worlds, with each succeeding world considering additional complications in the scene geometry.

The first world, i.e., *World I*, is *surface smoothness*. As shown in [12], we assume a simple world in which the scenes captured in a stereo pair contain only one object. On the surface of this object, we further assume that the 2-D distance function of the cyclopean coordinate system is everywhere continuous; so, a particular epipolar line has both a distance function  $D$  and a disparity function  $d$  which are also everywhere continuous. Because the relation between disparity and depth is known, we do not explicitly represent the depth, but rather the disparity in the derivation of the prior model. In the second world, i.e., *World II – Object Boundaries*, we assume a slightly more complicated world than *World I*: here we consider the possibility of more than one object in a scene. For this world the assumption is that the disparity function  $d$  is a sample path of the sum of a scaled Brownian motion process and a compound Poisson process with i.i.d., uniform random variables. In this way, we were able to consider multiple objects in a scene by introducing random variables which explicitly represented the discontinuities in disparity at the boundaries of objects. The third world is a more complicated one, which is the world for *Surface Slope and Creases*. In this world, not only do we consider more than one object in a scene, but we also consider that surfaces of objects may be steeply sloping and may have creases. We recommend [12] for further implementation details about the proposed model.

In the rest of this section, we present a two-level generative model that incorporates generic visual knowledge for dense stereo reconstruction introduced by [3]. In this work, the visual knowledge is represented by a dictionary of surface primitives including various categories of boundary discontinuities and junctions in parametric form. Given a stereo pair, we first compute a primal sketch representation which decomposes the image into a structural part for object boundaries and high intensity contrast represented by a 2D sketch graph, and a structureless part represented by Markov random field on pixels. Then we label the sketch graph and compute the 3D sketch (like a wire-frame) by fitting the dictionary of primitives to the sketch graph. The surfaces between the 3D sketches are filled in by computing the depth of the MRF (Markov random field) model on the structureless part using conventional stereo methods. These two levels interact closely with the primitives acting as boundary conditions for MRF, and the MRF propagating information between the primitives. The two processes maximize a Bayesian posterior probability jointly. [3] proposes an MCMC algorithm that simultaneously infers the 3D primitive types and parameters and estimates the depth (2.5D sketch) of the scene. The experiments show that this representation can infer the depth map with sharp boundaries and junctions for texture-less images, curve objects, and free-form shapes.

The overall dataflow of the algorithm is illustrated in Figure 8.5. Given a stereo pair of images, [3] first computes a primal sketch representation [81] which decomposes the image into two layers. (i) A structural layer for object boundaries and high intensity contrast represented by a 2D sketch graph, and (ii) a structureless layer represented by Markov random field on pixels. The sketch graph in the structural layer consists of a number of isolated points, line segments, and junctions which are considered vertices of different degrees of connection.

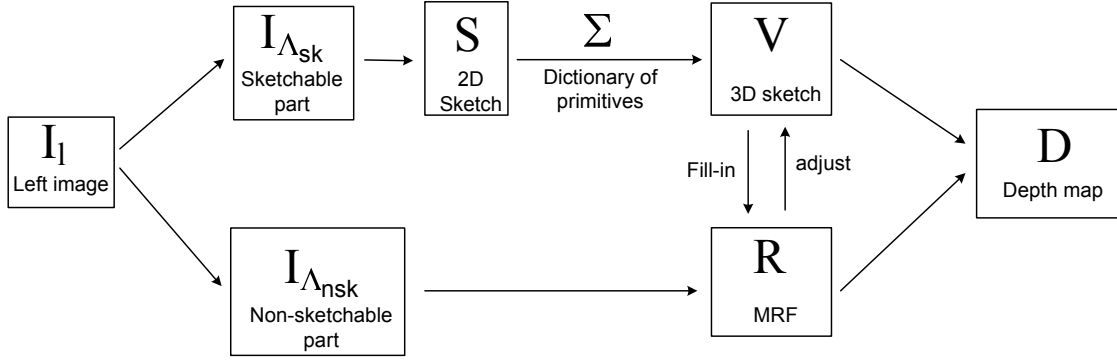


Figure 8.5: Reprinted with permission from [3]. The flow diagram of the algorithm.

[3] then studies the 3D structures for these points, line segments, and junctions and develops a dictionary for different configurations. The boundary primitives correspond to places where the depth map is not smooth, namely the boundaries between objects in the scene (first-order discontinuities) and the places where the surface normally experiences large changes in direction (second-order discontinuities). The curve primitives describe thin curves of different intensity from the background, and usually represent wire-like 3D objects such as thin branches of a tree or electric cables, etc. The point primitives represent feature points in the image that have reliable depth information. The valid combinations of these 3D primitives are summarized in a dictionary of junctions. Figure 8.8 and 8.9 show the dictionaries of line segments and junctions respectively. Each is a 3D surface primitive specified by a number of variables. The number of variables is reduced for degenerated (accidental) cases.

[3] adopts a probability model in a Bayesian framework, where the likelihood is described in terms of the matching cost of the primitives to images, while the prior has terms for continuity and consistency between the primitives, and a Markov Random Field that is used to fill in the depth information in the structureless areas. This Markov Random Field together with the labeling of the edges can be thought of as a Mixed Markov Model [58], in which the neighborhood structure of the MRF depends upon the types of the primitives, and changes dynamically during the computation.

The inference algorithm simultaneously finds the types of the 3D primitives, their parameters and the full depth map (2.5D sketch). To make up for the slowdown given by the long range interactions between the primitives through the MRF, the algorithm makes use of data driven techniques to propose local changes (updates) in the structureless areas.

The model proposed in [3] is different from other models existent in the literature in the close relationship between the MRF and the boundary primitives. The non-horizontal boundary primitives serve as control points for the MRF, while the horizontal primitives and the occluded sides of the primitives have their disparity determined by the MRF.

In the following subsections, we introduce more details about the work in [3]. We first present the representation, dictionary of primitives and junctions and probabilistic formulation in section 8.2.2, and then explain the algorithm details in section 8.2.3. Finally, the results obtained in [3] is shown in section 8.2.4.

## 8.2.2 Two Layer Representation

Given a stereo pair  $I_l, I_r$  of images, we would like to find the depth of all pixels in  $I_l$ . Assuming that the camera parameters are known, this is equivalent to finding for each pixel, the horizontal disparity that matches it to a corresponding pixel in  $I_r$ . Let  $D$  be the disparity map that needs to be inferred and  $\Lambda$  be the pixel lattice.

We assume the disparity map  $D$  is generally continuous and differentiable, with the exception of a

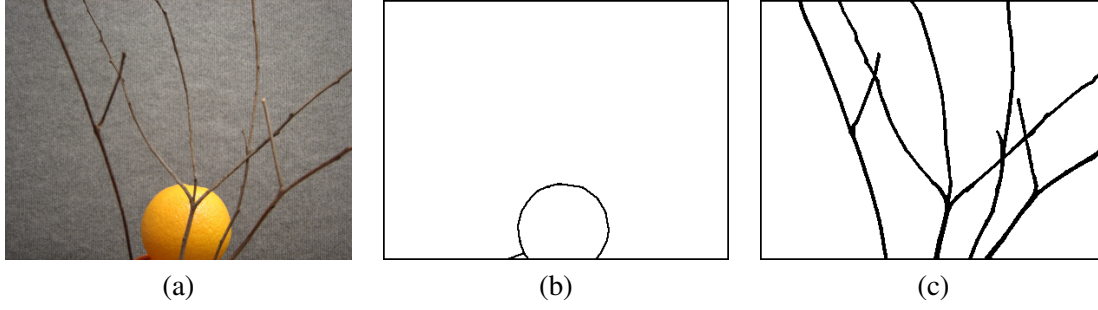


Figure 8.6: Reprinted with permission from [3]. The algorithm proposed in [3] starts from a two layer sketch representation. (a) input image, (b) region layer, (c) curve layer.

number of curves  $\Lambda_{sk}$  where the continuity or differentiability assumption does not hold. These curves are augmented with disparity values and are considered to form a 3D sketch  $D_s$  that acts as boundary conditions for the Markov Random Field modeling the disparity on  $\Lambda_{nsk} = \Lambda \setminus \Lambda_{sk}$ .

**A. The sketch layer - from 2D to 3D.** Assume that the places where the disparity is discontinuous or non-differentiable are among the places of intensity discontinuity. The intensity discontinuities are given in the form of a sketch  $S$  consisting of a region layer  $S_R$  and a curve layer  $S_C$ , as illustrated in Figure 8.6. The curve layer is assumed to occlude the region layer. These sketches can be obtained as in [81, 215]. The sketch edges are approximated with line segments  $S = \{s_i, i = 1, \dots, n_e\}$ . The segments that originated from the region layer  $s_i \in S_R$  will be named *edge segments* while the segments originating from the curve layer  $s_i \in S_C$  will be named *curve segments*.

Each edge segment  $s_i \in S_R$  from the region layer is assigned two 5-pixel-wide *edge regions*  $l_i, r_i$ , on the left and on the right of  $s_i$  respectively, as shown in Figure 8.7, left. Each curve segment  $s_j \in S_C$  is assigned a *curve region*  $r_j$  along the segment, of width equaling to the width of the curve, as shown in Figure 8.7, right. Denote the pixels covered by the edge and curve regions by  $\Lambda_R, \Lambda_C$  respectively.

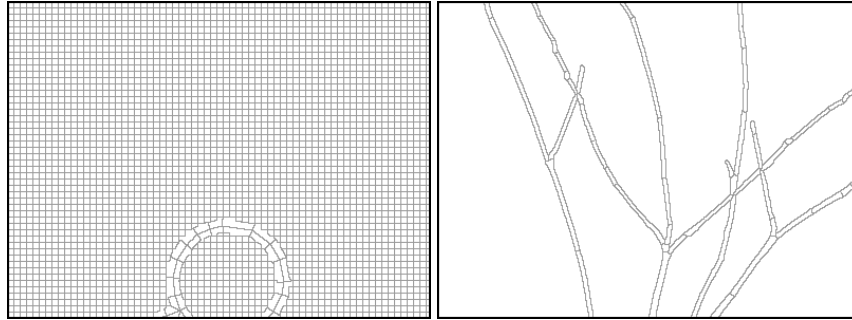


Figure 8.7: Reprinted with permission from [3]. Division of the image in Figure 8.6 into sketch primitives and 6x6 pixel square regions. Region layer (left) and curve layer (right).

Because of away from the places of discontinuity, the surfaces are in general very smooth and to reduce the dimensional of the problem, the pixels of  $\Lambda \setminus \Lambda_R$  are grouped into *square regions* of size  $6 \times 6$  pixels, by intersecting  $\Lambda \setminus \Lambda_R$  with a  $6 \times 6$  rectangular grid. Small regions at the boundary between the edge regions and the rectangular grid are merged to the edge regions. This way, the entire lattice  $\Lambda$  is divided into atomic regions that either are along the sketch  $S_C$ , or are on the rectangular grid, as shown in Figure 8.7. This structure allows the use of the thin plate spline model for the MRF and also enables implementation of good boundary conditions by the 3D primitives.

Then all line segments  $s_i \in S$  are augmented with parameters to become 3D sketch primitives, as shown

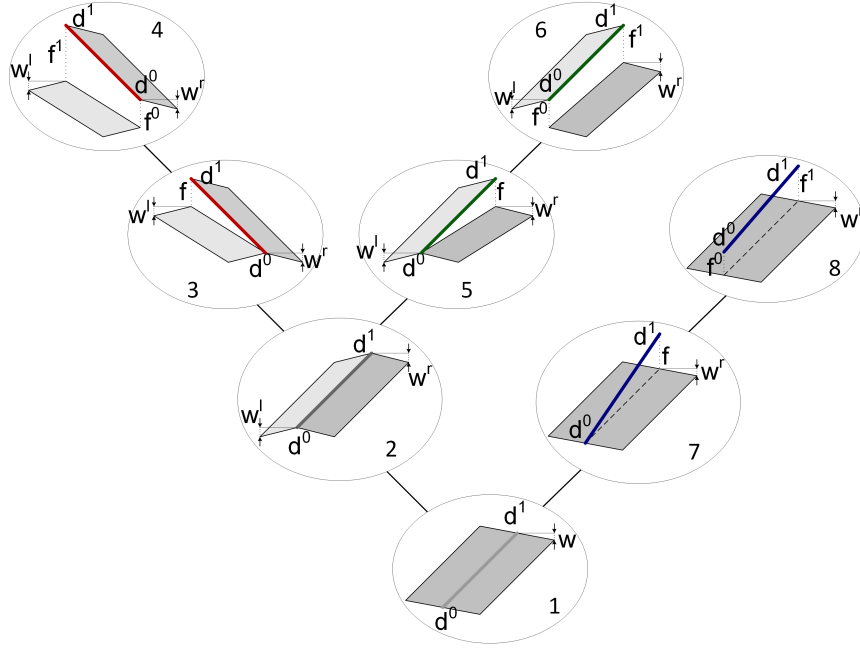


Figure 8.8: Reprinted with permission from [3]. Each sketch segment is augmented to a primitive from the following dictionary, ordered by generality.

in Figure 8.8. Depending on the type of segments they originated from, there are *boundary primitives* and *curve primitives*.

**B. A dictionary of primitives.** Let

$$V_1 = \{\pi_i = (s_i, [l_i, o_i^l], r_i, o_i^r, t_i, p_i, d_i, w_i[, f_i]), i = 1, \dots, n_e\} \quad (8.7)$$

be the set of all primitives, where the parameters in brackets might be missing, depending on the primitive type. The variables of each primitive are:

1. The edge segment  $s_i \in S_R$  or curve segment  $s_i \in S_C$
2. The left and right regions (wings)  $l_i, r_i$  in case of an edge segment, or the curve as a region  $r_i$  in case of a curve segment.
3. an occlusion label  $o_i^l, o_i^r$  for each of the regions  $l_i, r_i$ , representing whether the region is occluded (value 0) or not (value 1).
4. The label  $t_i = t(\pi_i) \in \{1, \dots, 8\}$  indexing the type of the primitive from the *primitive dictionary* with the restriction that edge segments  $s_i \in S_R$  can only be assigned types from  $\{1, \dots, 6\}$  while curve segments  $s_i \in S_C$  can only be assigned types from  $\{1, 7, 8\}$ . These types are illustrated in Figure 8.8.
  - Type 1 represents edges or curves that are on the surface of the objects.
  - Type 2 represents first order discontinuities, i.e. places where the surface is continuous but the normal is discontinuous.
  - Types 3, 4, 5, 6 represent occluding edges where the occluded surface is on the left (types 3, 4) or on the right (types 5, 6) of the edge.
  - Types 7, 8 represent 3D curves, either connected with one end to the surface behind, or totally disconnected.

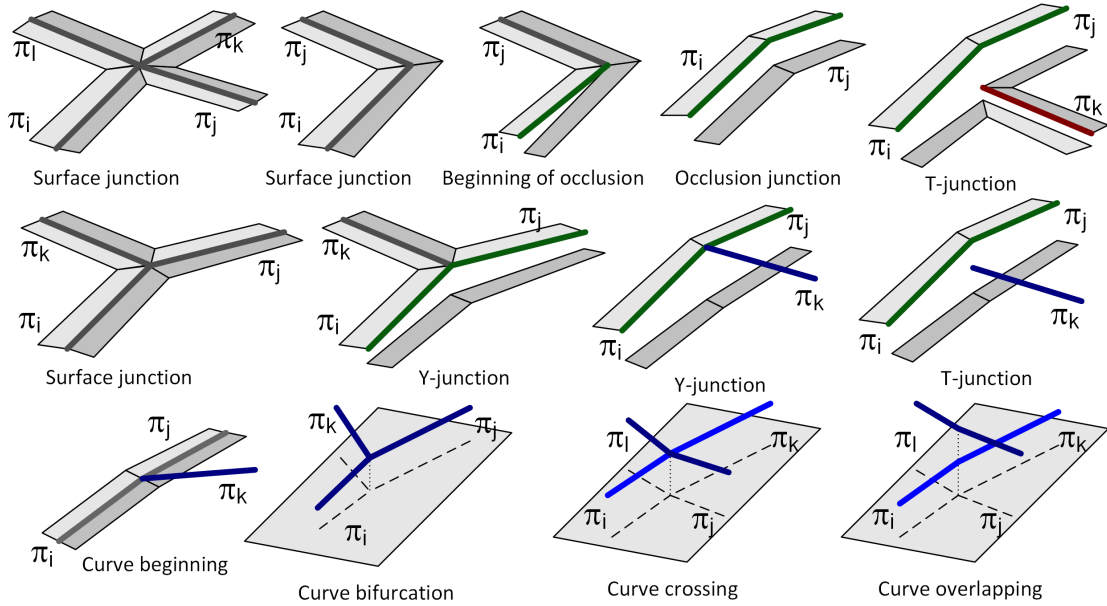


Figure 8.9: Reprinted with permission from [3]. These are the main types of junctions between boundary and curve primitives.

5. A label  $p_i$  specifying whether this primitive is a control point (value 1) of the thin plate spline or not (value 0). All horizontal edges have  $p_i = 0$  at all times.
6. The disparities  $d_i = d(\pi_i) = (d_i^0, d_i^1)$  at the endpoints of the segment or the disparity  $d_i = d(\pi_i)$  at the center of the segment if the segment is short (less than 6 pixels long).
7. The left and right control arm  $w_i = w(\pi_i) = (w_i^l, w_i^r)$  representing the slope of the disparity map  $D$  in the direction perpendicular to the line segment.
8. For types 3-6, the disparity  $f_i = f(\pi_i) = (f_i^0, f_i^1)$  of the occluded surface at the ends of the segment, or the disparity  $f_i = f(\pi_i)$  at the center of the edge segment if the segment is short (less than 6 pixels long).

Each of the regions  $l_i, r_i$  of the primitive  $\pi_i = (s_i, [l_i, o_i^l], r_i, o_i^r, t_i, p_i, d_i, w_i[, f_i])$  is assigned a matching cost where for each pixel  $v \in r_i$ , the disparity  $d_v(\pi_i)$  is the linear interpolation based on the parameter  $d$  representing the disparity at the ends of the region, in the assumption that  $w = 0$ .

$$c(r_i, d) = \begin{cases} 0 & \text{if } r_i \text{ intersects the curve sketch } S_C \\ \sum_{v \in r_i} |\mathbf{I}_l(v) - \mathbf{I}_r(v - d_v(\pi_i))| & \text{if } o_i^r = 1 \\ \alpha & \text{else} \end{cases} \quad (8.8)$$

Then the matching cost of the primitive  $\pi_i$  is

$$c(\pi_i) = c(r_i, [l_i], t_i, d_i, [f_i]) = \begin{cases} c(r_i, d_i) & \text{if } t_i = 7, 8, 1(\text{curve}) \\ c(l_i, d_i) + c(r_i, d_i) & \text{if } t_i = 2, 1(\text{region}) \\ c(l_i, f_i) + c(r_i, d_i) & \text{if } t_i = 3, 4 \\ c(l_i, d_i) + c(r_i, f_i) & \text{if } t_i = 5, 6 \end{cases} \quad (8.9)$$

The primitives form a graph by the natural adjacency relation between the underlying edge segments.

**C. Modeling junctions between the primitives.** To increase the model accuracy, the junction points of two or more primitives are modeled. Similar to [194], certain types of possible junctions depending on the degree (number of primitives) of the junction are introduced below and illustrated in Figure 8.9. These junctions include:

- Junctions of 2 boundary primitives have three main types: Surface junctions, beginning of occlusion and occlusion junctions.
- Junctions of 3 boundary primitives have three main types: Surface junctions, Y-junctions and T-junctions.
- Junctions of 4 or more boundary primitives are accidental and are assumed to be all surface junctions.
- No junctions between one or two curve primitives and one boundary primitive.
- Junctions of 1 curve primitive with two boundary primitives have three main types: curve beginning, Y-junctions and T-junctions.
- Junctions of 2 curve primitives have only one type.
- Junctions of 3 curve primitives have only one type, namely bifurcation.
- Junctions of 4 curve primitives have two types, namely curve crossing or curve overlapping. In both cases, the opposite primitives can be seen as part of the same 3D curve.

Let  $J = \{\phi_i = (t, k, \pi_{i_1}, \dots, \pi_{i_k}), \pi_{i_1}, \dots, \pi_{i_k} \in V_1, i = 1, \dots, n_J\}$  be the set of junctions, each containing the list of primitives that are adjacent to it. The variable  $t$  is the junction type and restricts the types of the primitives  $\pi_{i_1}, \dots, \pi_{i_k}$  to be compatible to it.

Each junction  $\phi_i \in J$  imposes a prior model that depends on the junction type, and the types and directions of the 3D primitives  $\pi_{i_1}, \dots, \pi_{i_k}$  meeting in this junction. This prior is composed of a 3D geometric prior on the primitives and a 2D occurrence prior of each particular junction type.

Thus

$$P(\phi) \propto P(\pi_{i_1}, \dots, \pi_{i_k} | t, \phi^{2D}) P(\phi^{2D}, t) = P(\phi^{3D} | t, \phi^{2D}) P(t | \phi^{2D}) \quad (8.10)$$

since the 2d geometry  $\phi^{2D}$  of the junction is fixed.

We will now discuss  $P(\phi^{3D} | t, \phi^{2D})$  for each junction type. To simplify the notation, we define two continuity priors:

$$\begin{aligned} p_c(\pi_i, \pi_j) &= \frac{1}{Z_c} \exp(-\beta_c |d_i^\phi - d_j^\phi|^2) \\ p_s(\pi_i, \pi_j) &= \frac{1}{Z_s} \exp[-\beta_c |d_i^\phi - d_j^\phi|^2 - \beta_s (|d_i^{\bar{\phi}} - 2d_i^\phi + d_j^{\bar{\phi}}|^2 - |d_i^{\bar{\phi}} - 2d_j^\phi + d_j^{\bar{\phi}}|^2)] \end{aligned} \quad (8.11)$$

where  $d_i = (d_i^\phi, d_i^{\bar{\phi}})$  is the disparity of the primitive  $\pi_i$ , with  $d_j^\phi$  being the disparity at the junction  $\phi$  endpoint.

1) All the surface junctions of 3 or more boundary primitives and the curve bifurcation or crossing have a prior that prefers the same disparity for all primitives meeting at this junction.

$$P(\phi^{3D} | t, \phi^{2D}) = \frac{1}{Z_1} \prod_{\pi_j, \pi_k \in \phi} p_c(\pi_j, \pi_k) \quad (8.12)$$

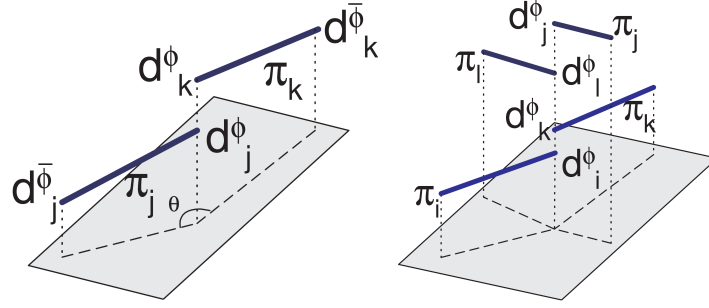


Figure 8.10: Reprinted with permission from [3]. Left: The prior of the junction between two boundary or curve primitives depends on the angle  $\theta$  between the primitives. Right: The prior of the curve overlapping junction encourages continuity of each pair of opposite curves.

2) The prior of junctions of two boundary or two curve primitives depends on the angle  $\theta$  between the primitives at the junction.

$$P(\phi^{3D}|t, \phi^{2D}) = \frac{1}{Z_2} \begin{cases} p_c(\pi_j, \pi_k) & \text{if } |\theta - \pi| > \pi/6 \\ p_s(\pi_j, \pi_k) & \text{else} \end{cases} \quad (8.13)$$

as shown in Figure 8.10, left.

3) For the curve overlapping junction involving four curve primitives, the prior is defined in terms of the continuity of each pair of opposite curves.

$$P(\phi^{3D}|t, \phi^{2D}) = p_s(\pi_i, \pi_k)p_s(\pi_j, \pi_l) \quad (8.14)$$

as shown in Figure 8.10, right.

4) For the Y-junctions of 3 boundary primitives and for the curve beginning, the prior encourages all three primitives to be adjacent, and the primitives  $\pi_i, \pi_j$  (refer to Figure 8.9) to have a good continuation as in case 2).

$$P(\phi^{3D}|t, \phi^{2D}) = \frac{1}{Z_4} p_s(\pi_i, \pi_j) \prod_{\pi_u, \pi_v \in \phi, \{u,v\} \neq \{i,j\}} p_c(\pi_u, \pi_v) \quad (8.15)$$

5) For the T-junctions, the prior encourages continuity of the occluding edge.

$$P(\phi^{3D}|t, \phi^{2D}) = p_s(\pi_i, \pi_j) \quad (8.16)$$

Since the disparity space of each primitive is discretized, the normalizing constant for each junction can be computed effectively.

The prior  $P(t|\phi^{2D})$  can be learned from hand labeled data, independently for each degree (number of primitives)  $k$  of the junction.

Based on the matching cost, a saliency map

$$\psi_{\pi_i}(d, [f]) = \exp(-c(r_i, [l_i], t_i, d, [f])/10) \quad (8.17)$$

towards all possible values of  $d, f$  is computed for each primitive  $\pi_i \in V_1$ . This information will be used to find the disparities  $d_i$  of the sketch primitives.

We also compute a saliency map towards the three main types of boundary primitives, namely surface (types 1, 2), occluding left (types 3, 4), occluding right (types 5, 6), based on the feature

$$\xi(\pi_i) = \frac{\min_d c(l_i, d)}{|l_i|} - \frac{\min_d [c(l_i, d) + c(r_i, d)]}{|l_i| + |r_i|} \quad (8.18)$$



which measures how well both wings of the primitive fit the same disparity, as compared to the left wing alone.

From hand labeled data, we obtained histograms  $H_{12}, H_{34}, H_{56}$  of the values of  $\xi$  for each of the three main types of boundary primitives. We fit these histograms with Gaussians to even out the small amount of training data and eliminate the need for histogram bins. From here we obtain a likelihood  $L_\pi(t)$  towards the three main types of boundary primitives.

$$L_\pi(t) = \begin{cases} 60e^{-\xi^2/2} & \text{if } t = 1, 2 \\ 4.4e^{-(\xi+1.18)^2/1.42} + 3.67e^{-(\xi+8.21)^2/6} & \text{if } t = 3, 4 \\ 9.18e^{-(\xi-0.58)^2/0.87} + 3.06e^{-(\xi-7.36)^2/7.5} & \text{if } t = 5, 6 \end{cases} \quad (8.19)$$

Using the intensity-driven likelihood for the boundary primitives, we construct a likelihood, driven simultaneously by the image intensity and the geometry (relative position of primitives), for each junction  $\phi = \{\pi_1, \dots, \pi_k\}$ :

$$L_\phi(t) = P(\phi)L_{\pi_1}(t_1)\dots L_{\pi_k}(t_k) \quad (8.20)$$

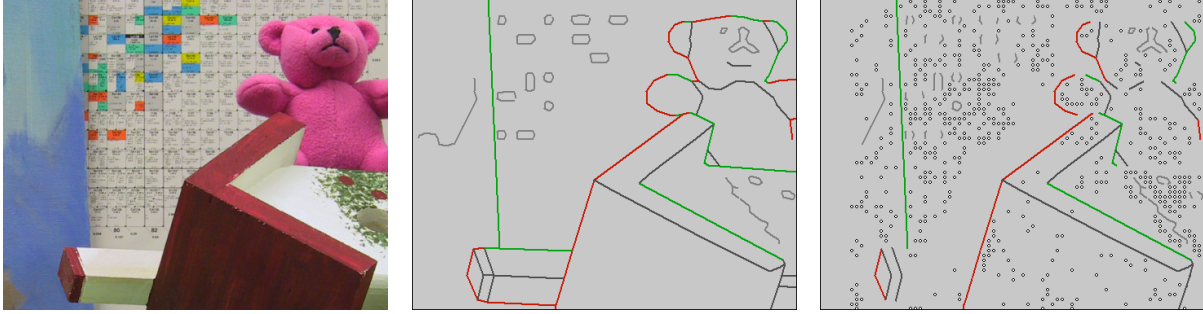


Figure 8.11: Reprinted with permission from [3]. Left image of a stereo sequence, the graph labeling and the control points (point and boundary primitives) of the thin plate spline.

**D. The free-form layer.** The primitives  $\pi \in V_1$  discussed in the previous section are elongated primitives corresponding to line segments, so they can be considered of dimension 1. Other sketch primitives that are involved in the free form layer are the zero dimensional primitives corresponding to feature points with reliable disparity information, i.e. *point primitives*. These primitives are a subset of the rectangular atomic regions, and together with the one dimensional boundary primitives are the control points of the thin plate spline. The curve primitives are not involved in the MRF computation.

Let  $R$  be the set of all rectangular atomic regions. For each region  $r \in R$ , we compute a saliency map

$$\rho_r(d) \propto \exp\left(-\sum_{v \in r} |\mathbf{I}_l(v) - \mathbf{I}_r(v-d)|/10\right) \quad (8.21)$$

to all possible disparities  $d \in [d_{\min}, d_{\max}]$ . Then the square regions

$$R = \{r_i = (d_i, o_i, p_i, \mu_i, \sigma_i^2), i = 1, \dots, n_r\} \quad (8.22)$$

have the following parameters:

1. the disparity  $d_i = d(r_i)$  of the center of the region
2. a label  $o_i$  specifying whether this region is occluded (value 0) or not (value 1).

3. a label  $p_i = p(r_i) \in \{0, 1\}$  representing whether the region is a point primitive (i.e. control point for the thin plate spline) or not.
4. the mean  $\mu_i$  and variance  $\sigma_i^2$  of the saliency map  $\rho_{r_i}$ .

Following [12], all regions (edge regions, curve regions, and square regions) will have their occlusion labels deterministically assigned based on the disparities of the boundary and curve primitives. For example, for an occlusion primitive  $\pi_i$  of type 4, the left region  $l_i$  and other regions horizontally to the left of the edge at horizontal distance less than the disparity difference between the right and left wings of  $\pi_i$  will be labeled as occluded.

The matching cost for each region  $r_i \in R$  is

$$c(r_i) = \begin{cases} \alpha & \text{if } o_i = 0 \\ \sum_{v \in r_i} |\mathbf{I}_l(v) - \mathbf{I}_r(v - d_i)| & \text{if } o_i = 1 \end{cases} \quad (8.23)$$

The set of point primitives is denoted by

$$V_0 = \{r_i \in R, s_i = 1\}. \quad (8.24)$$

In Figure 8.11 are shown the labeled graph, i.e. primitive types (middle), and the point and boundary primitives that act as control points for the  $\Lambda_{nsk}$  part (right). The depth and disparity maps obtained this way are shown in Figure 8.15. Observe that the horizontal edges are not control points.

The dense disparity map  $D$  is obtained from  $V_1$  and  $R$  by interpolation. By using the boundary primitives to model the places of discontinuity, the obtained disparity map has crisp discontinuities at the object boundaries and is smooth everywhere else, as shown in Figure 8.15.

**E. Bayesian formulation.** We formulate our model using the Bayes rule:

$$P(V_1, R | \mathbf{I}_l, \mathbf{I}_r) \propto P(\mathbf{I}_l | \mathbf{I}_r, V_1, R) P(R - V_0 | V_0, V_1) P(V_0, V_1) \quad (8.25)$$

The likelihood  $P(\mathbf{I}_l | \mathbf{I}_r, V_1, R)$  is expressed in terms of the likelihood  $L_{\pi_i}(t_i)$  and matching cost  $c(r_j)$  of the sketch primitives.

$$P(\mathbf{I}_l | \mathbf{I}_r, V_1, R) \propto \prod_{i=1}^{n_e} L_{\pi_i}(t_i) \exp[-\sum_{r_j \in R} c(r_j)] \quad (8.26)$$

The prior

$$P(R - V_0 | V_0, V_1) \propto \exp[-E_c(R) - \beta_b E_b(R, V_1)]$$

is defined in terms of the energy of the soft control points:

$$E_c(R) = \sum_{r_j \in V_0} (d_j - \mu_j)^2 / 2\sigma_j^2$$

and the thin plate bending energy:

$$E_b(R, V_1) = \sum_{(x,y) \in G} [d_{xx}^2(x, y) + 2d_{xy}(x, y)^2 + d_{yy}^2(x, y)],$$

which is computed on a  $6 \times 6$  grid  $G$  containing the centers of all the square regions and neighboring grid points on the boundary primitives. For example, if the point  $(x, y) \in G$  is the center of  $r_j \in R$  and  $r_N, r_{NW}, r_W, r_{SW}, r_S, r_{SE}, r_E, r_{NE}$  are the 8 neighbors of  $r_j$ , then

$$\begin{aligned} d_{xx}(x, y) &= d_W - 2d_j + d_E \\ d_{yy}(x, y) &= d_N - 2d_j + d_S \\ d_{xy}(x, y) &= (d_{NE} + d_{SW} - d_{NW} - d_{SE})/4 \end{aligned}$$

Similar terms in the bending energy  $E_b(R, V_1)$  can be written for cases where one or many of the neighbors are boundary primitives. However, there are no terms involving the left and right atomic regions  $l_i, r_i \in \pi_i$  belonging to the same edge primitive  $\pi_i$ .

The prior  $P(V_0, V_1) = P(V_0)P(V_1)$  assumes a uniform prior on  $V_0$  while  $P(V_1)$  is defined in terms of the junction priors  $P(\phi_i)$  defined above,  $P(V_1) = \prod_{\phi_i \in J} P(\phi_i)$ .

### 8.2.3 The Inference Algorithm

In our problem formulation, there are two types of variables, discrete and continuous. The discrete variables are

$$\Delta = V_1^d \cup R^d \quad (8.27)$$

consisting of  $V_1^d = \{(t(\pi), o^l(\pi), o^r(\pi), p(\pi)), \forall \pi \in V_1\}$  and  $R^d = \{(s(r), o(r), p(r)), \forall r \in R\}$ . All other variables are continuous variables, namely  $V_1^c = V_1 \setminus V_1^d$  and  $R^c = R - R^d$ , and can be divided into the boundary conditions

$$\Gamma = V_0^c \cup \{d(\pi), \forall \pi \in V_1, p(\pi) = 1\}$$

and the fill-in variables

$$\Psi = \{([w(\pi)], [f(\pi)]), \forall \pi \in V_1\} \cup \{d(\pi), \forall \pi \in V_1, p(\pi) = 0\} \cup R^c - V_0^c.$$

The posterior probability can then be written as

$$p(V_1, R | \mathbf{I}_l, \mathbf{I}_r) = p(\Delta, \Gamma, \Psi | \mathbf{I}_l, \mathbf{I}_r) \quad (8.28)$$

In a MAP formulation, our algorithm needs to perform the following three tasks:

1. Reconstruct the 3D sketch to infer the parameters  $\Gamma$  of the primitives.
2. Label the primitive graph to infer the discrete parameters  $\Delta$ , i.e. associates the primitives with the appropriate types. This represents the detection of surface boundaries and of the feature points of the image.
3. Perform "fill in" of the remaining parts of the image, using the MRF and  $\Gamma, \Delta$  as boundary conditions, to infer  $\Psi$  and obtain a dense disparity map  $D$ .

The algorithm will proceed as follows. In an initialization phase, the first two steps will be performed to compute an approximate initial solution. Then steps 2) and 3) will be performed to obtain the final result.

**A. Initialization.** Initializing the system purely based on the local depth  $\psi_\pi$  and likelihood  $L_\pi(t)$  information existent at the primitives  $\pi \in V_1$  results in an inconsistent initial solution which is valid only at places with reliable local depth information, as shown in Figure 8.12, left.

A major improvement can be achieved by using the junction prior  $P(\phi)$  that has been defined in Section 8.2.2C, which provides a way to propagate depth information quickly along the edges of the sketch, from the places where it is available. This is why we use an approximation of the posterior probability that only takes into account the matching cost of the edge regions  $\pi_i \in V_1$  and the junction prior.

$$P(V_1 | \mathbf{I}_l, \mathbf{I}_r) \propto \prod_{i=1}^{n_e} L_{\pi_i}(t_i) \prod_{\phi_i \in J} P(\phi_i) \quad (8.29)$$

At this stage, the variables that highly depend on the thin plate spline prior will be assigned some default values. Thus, the wing parameters  $w_i, \forall \pi_i \in V_1$  will be assigned value 0 (i.e. all wings will be horizontal), while the occlusion labels  $o_i$  will be assigned value 1 (unoccluded).

The initialization algorithm alternates the following MCMC steps:

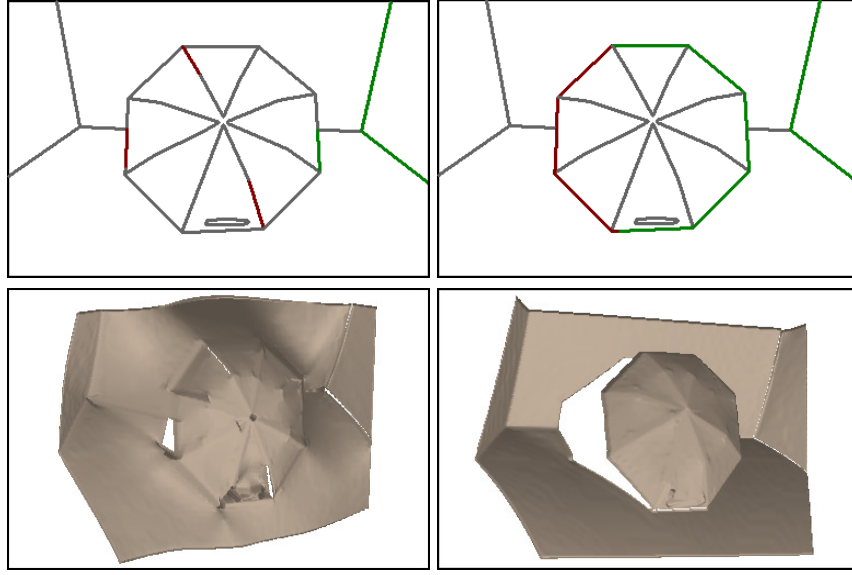


Figure 8.12: Reprinted with permission from [3]. Left: An initialization purely based on local information is not satisfactory. Right: By propagating the junction priors along the sketch, a much better initialization can be quickly obtained.

- a single node move that changes one variable  $d_i$  at a time.
- a move that simultaneously shifts all  $d_i$  at the same junction  $\phi$  by the same value. This move is capable of adjusting the disparity of primitives at a junction at times when changing the disparity of only one primitive will be rejected because of the continuity prior.
- a labeling move as described in the MCMC algorithm below, which proposes a new labeling for a set of primitives and junctions. The move is accepted using the Metropolis-Hastings method based on the posterior probability from Eq. (8.29).

The algorithm is run for  $10|V_1|$  steps and obtains the initialization result shown in Figure 8.12, right in about 10 seconds. The initialization algorithm is very fast because the fill-in of the interior pixels is not performed, eliminating the expensive MRF computation.

The 3D reconstruction of the curve primitives is performed separately in a similar manner. The labeling move is much simpler, since the curve primitives basically accept two labels, surface/non-surface. The rectangular regions with low matching cost and small variance (less than 1) will initially be labeled as control points for the thin plate spline.

**B. Updating the fill-in variables  $\Psi$ .** Observe that in our formulation of the energy, if  $\Delta, \Gamma$  are fixed, the conditional  $-\log(P(\Psi|\Delta, \Gamma))$  is a quadratic function in all the variables  $\Psi$ , so it can be minimized analytically. This implies that  $\Psi$  can be regarded as a function on  $\Delta, \Gamma$ ,  $\Psi = \Psi(\Delta, \Gamma)$ . This restricts the problem to maximizing the probability  $P(\Delta, \Gamma, \Psi(\Delta, \Gamma)|\mathbf{I}_l, \mathbf{I}_r)$ , of much smaller dimensionality.

Inside each of the regions  $C$  bounded by the control point sketch primitives, the variables depend only on the control points inside and on the boundary of this region. So the computation can be localized to each of these regions independently, as shown in Figure 8.13. Additional speedups can be obtained following the approximate thin plate spline methods from [42].

Observe that the update can affect some non-control point edges, such as the horizontal edges from Figure 8.13.

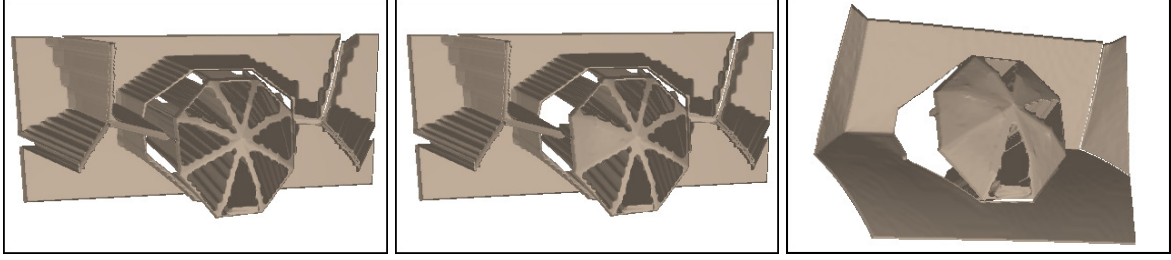


Figure 8.13: Reprinted with permission from [3]. The fill-in can be restricted to the connected components bounded by control point boundary primitives. In a few steps, the initial 3D reconstruction before graph labeling is obtained. Shown are the 3D reconstructions after 0,1, and 5 connected components have been updated. The horizontal edges change the disparity at the same time with the interior, because they are not control points.

For each such region  $C$ , we define relative labels  $l_C$  of the edges adjacent to  $C$  that only take into account the side of the edge that belongs to  $C$ . For example, an occluding edge type 4 and an edge of type 1 will have the same label relative to the region  $C$  containing the right wing of the edge. Using these relative labels, we reduce the computation expense by defining the energy of the region

$$E(C, l_C) = E_c(C) + \mu_b E_b(C) + \sum_{r \in C \cap R} c(r) \quad (8.30)$$

The full posterior probability can be recovered from the energy of the regions and the junction prior:

$$P(V_1, R | \mathbf{I}_l, \mathbf{I}_r) \propto \prod_{i=1}^{n_e} L_{\pi_i}(t_i) \exp[-\sum_C E(C, l_C)] \prod_{\phi \in J} P(\phi) \quad (8.31)$$

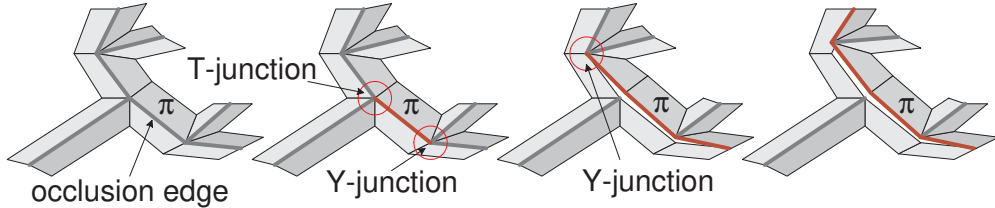


Figure 8.14: Reprinted with permission from [3]. Each graph labeling move changes the types of a set of primitives in a consistent manner. First a primitive  $\pi$  is chosen and its type is sampled from the likelihood  $L_{\pi}(t)$ , then the adjacent junctions change their type conditional on the chosen type of  $\pi$ , which in turn determine the types of the other primitives of the junctions, etc. The labeling move is accepted based on the Metropolis-Hastings method. Illustrated is the left side of the umbrella image.

**C. The MCMC optimization algorithm.** After the initialization, the 3D sketch variables  $\Gamma = \Gamma_0$  will be fixed. The algorithm will only update the primitive types  $\Delta$  and the fill-in variables  $\Psi$ .

To maximize  $P(\Delta, \Gamma_0, \Psi(\Delta, \Gamma_0) | \mathbf{I}_l, \mathbf{I}_r)$  we will use a Markov chain Monte Carlo algorithm that will sample  $P(\Delta, \Gamma_0, \Psi(\Delta, \Gamma_0) | \mathbf{I}_l, \mathbf{I}_r)$ , and this way obtain the most probable solutions.

At each step, the algorithm proposes, as shown in Figure 8.14, new types for a set of primitives  $N$  and junctions  $J$  in one move, as follows:

1. Grow a set  $N$  of primitives as follows:
  1. Choose a random non-horizontal primitive  $\pi$ .
  2. Initialize  $N = \{\pi\}$  and  $J = \{\phi_1, \phi_2\}$  where  $\phi_1, \phi_2$  are the two junctions adjacent to  $\pi$ .
  3. Sample the primitive type  $t(\pi)$  from the local likelihood  $L_{\pi}(t)$ .

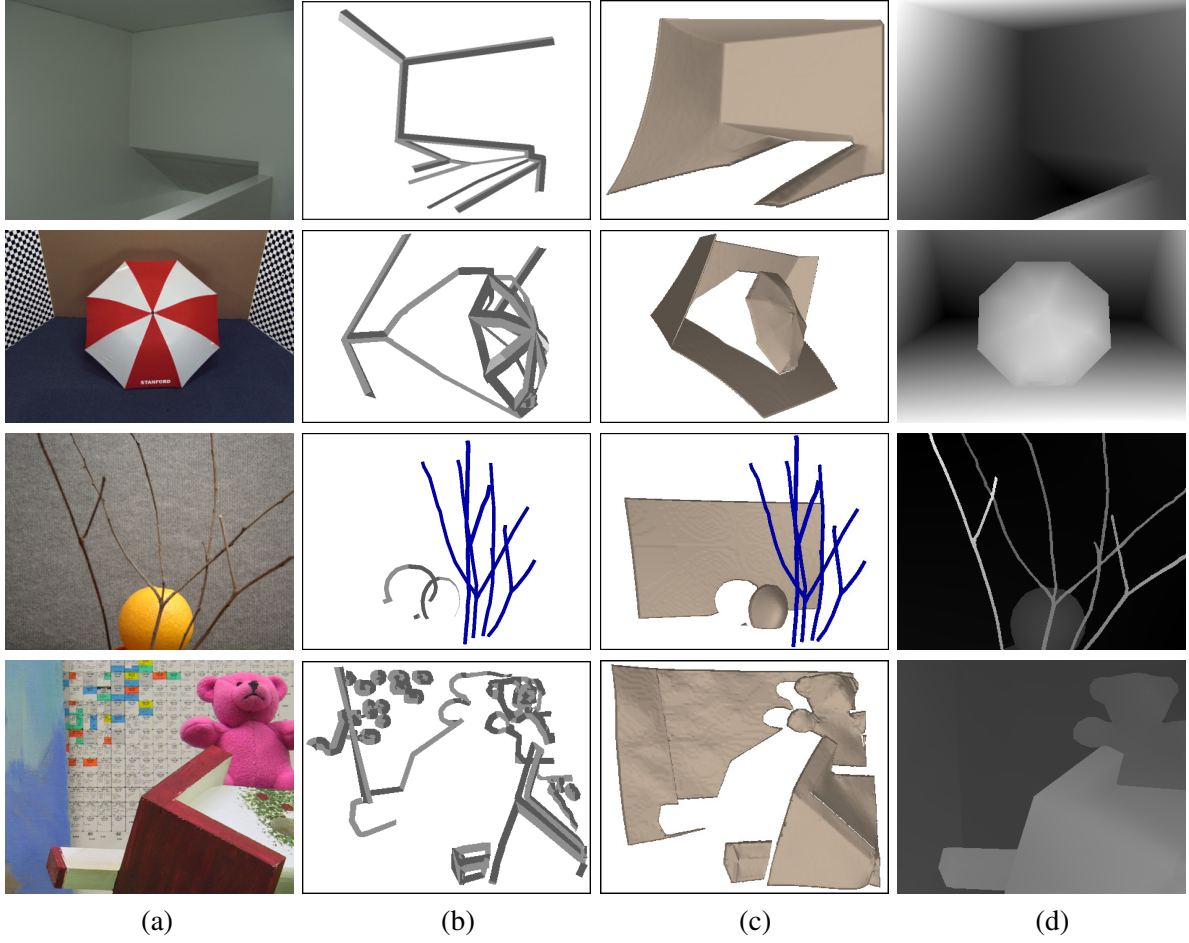


Figure 8.15: Reprinted with permission from [3]. Results obtained using our method. (a) left image of the stereo pair, (b) 3D sketch using the primitives, (c) 3D depth map, (d) disparity map.

4. Sample the type of  $\phi \in J$  from  $L_\phi(t)$ , conditional on the primitive type  $t(\pi)$ .  
This determines the types of all primitives of  $N_n = \{\pi' \notin N, \pi' \sim \phi \text{ for some } \phi \in J\}$ , where  $\pi \sim \phi$  means  $\pi$  is adjacent to  $\phi$ .
5. Set  $N \leftarrow N \cup N_n$ .
6. Initialize  $J_n = \emptyset$ .
7. For each  $\pi \in N_n$ , pick the adjacent junction  $\phi \notin J$ . If  $\pi$  changed its type at step 4, set  $J_n \leftarrow J_n \cup \{\phi\}$ , else set  $J_n \leftarrow J_n \cup \{\phi\}$  with probability 0.5.
8. Set  $J \leftarrow J \cup J_n$ .
9. Repeat steps 4-8 for each  $\pi \in N_n$  and each  $\phi \in J_n, \pi \sim \phi$ .
2. Update the fill-in variables  $\Psi(\Delta, \Gamma)$  for the connected components  $C$  where it is necessary.
3. Accept the labeling move based on the full posterior probability, computed using eq. (8.31).

## 8.2.4 Example Results

Experiments are presented in Figure 8.15 where four typical images for stereo matching are shown. The first two have textureless surfaces and the most information is from the surface boundaries. The fourth image has curves (twigs). For these images, it is not a surprise to see that the graph cut method with simple MRF models on pixels produces unsatisfactory results. The second and fourth images have free-form surfaces with or without textures from [137]) and [195]. On the teddy-bear sequence, the percentage of pixels with

error at least 1, as compared to the ground truth, is 3.3%. In comparison, the Graph Cuts result observes a 7.3% error rate. We have also shown the interactions of the two layers in Figure 8.13 and the effects of sketch labeling in Figure 8.14.

### 8.3 2.5D Sketch from Primal Sketch — Shape from Shading

In this section, we first briefly introduce the Lambertian model and the classic Horn's shape-from-shading, or in short SFS, work. Then we briefly present a two-level generative model for representing the images and surface depth maps of drapery and clothes.

In computer vision, the techniques to recover shape are called shape-from-X techniques, where X can be shading, stereo, motion, texture, etc. Shape-from-shading (SFS) deals with the recovery of shape from a gradual variation of shading in the image. Artists have long exploited lighting and shading to convey vivid illusions of depth in paintings. It is essential to study how the images are formed in order to solve the SFS problem. A simple model of image formation is the Lambertian model, in which the gray level at a pixel in the image depends on the light source direction and the surface normal. In SFS, given a gray level image, the aim is to recover the light source and the surface shape at each pixel in the image. However, real images do not always follow the Lambertian model. Even if we assume Lambertian reflectance and known light source direction, and if the brightness can be described as a function of surface shape and light source direction, it is still not straightforward. The reason is that if the surface shape is described in terms of the surface normal, we have a linear equation with three unknowns, and if the surface shape is described in terms of the surface gradient, we have a nonlinear equation with two unknowns. Therefore, finding a unique solution to SFS is difficult; it requires additional constraints.

Shading plays an essential role in the human perception of surface shape. Researchers in human vision have attempted to understand and simulate the mechanisms by which our eyes and brains use the shading information to recover the 3D shapes. The extraction of SFS by the visual system is also strongly affected by stereoscopic processing. Barrow and Tenenbaum discovered that it is the line drawing of the shading pattern that seems to play a central role in interpreting shaded patterns [11]. Mingolla and Todd's study of the human visual system based on the perception of solid shape [158] indicated that the traditional assumptions in SFS – Lambertian reflectance, known light source direction, and local shape recovery – are not valid from a psychological point of view. From the above discussion, one can observe that the human visual system uses SFS differently than computer vision does typically.

In the 1990s, Horn et al. [98] discovered that some impossibly shaded images exist, which could not be shading images of any smooth surface under the assumption of uniform reflectance properties and lighting. For this kind of image, SFS will not provide a correct solution, so it is necessary to detect impossibly shaded images.

SFS techniques can be divided into four groups: minimization approaches, propagation approaches, local approaches, and linear approaches. Minimization approaches obtain the solution by minimizing an energy function. Propagation approaches propagate the shape information from a set of surface points (e.g., singular points) to the whole image. Local approaches derive shape based on the assumption of surface type. Linear approaches compute the solution based on the linearization of the reflectance map.

One of the earlier minimization approaches, which recovered the surface gradients, was by Ikeuchi and Horn [106]. Since each surface point has two unknowns for the surface gradient, and each pixel in the image provides one gray value, we have an underdetermined system. To overcome this, they introduced two constraints: the brightness constraint and the smoothness constraint. The brightness constraint requires that the reconstructed shape produce the same brightness as the input image at each surface point, while the smoothness constraint ensures a smooth surface reconstruction. The shape was computed by minimizing an



energy function which consists of the above two constraints. In general, the shape at the occluding boundary was given for the initialization to ensure a correct convergence. Since the gradient at the occluding boundary has at least one infinite component, the stereographic projection was used to transform the error function to a different space. Additionally, Brooks and Horn [22] minimized the same energy function in terms of the surface normal using these two constraints. For further improvements, Frankot and Chellappa [53] enforced integrability in Brooks and Horn’s algorithm to recover integrable surfaces (surfaces for which  $z_{xy} = z_{yx}$ ). Surface slope estimates from the iterative scheme were expressed in terms of a linear combination of a finite set of orthogonal Fourier basis functions. The enforcement of integrability was done by projecting the nonintegrable surface slope estimates onto the nearest (in terms of distance) integrable surface slopes. This projection was fulfilled by finding the closest set of coefficients which satisfy integrability in the linear combination. Their results showed improvements in both accuracy and efficiency over Brooks and Horn’s algorithm [22]. Later, Horn also [101] replaced the smoothness constraint in his approach with an integrability constraint. The major problem with Horn’s method is its slow convergence. Szeliski [208] sped it up using a hierarchical basis preconditioned conjugate gradient descent algorithm. Based on the geometrical interpretation of Brooks and Horn’s algorithm, Vega and Yang [218] applied heuristics to the variational approach in an attempt to improve the stability of Brooks and Horn’s algorithm.

As for traditional propagation approaches, Horn proposed the characteristic strip method [99]. A characteristic strip is a line in the image along which the surface depth and orientation can be computed if these quantities are known at the starting point of the line. Horn’s method constructs initial surface curves around the neighborhoods of singular points (singular points are the points with maximum intensity) using a spherical approximation. The shape information is propagated simultaneously along the characteristic strips outward, assuming no crossover of adjacent strips. The direction of characteristic strips is identified as the direction of intensity gradients. To get a dense shape map, new strips have to be interpolated when neighboring strips are not close to each other.

Examples for local approaches are [177] and [131]. Pentland’s local approach [177] recovered shape information from the intensity and its first and second derivatives. He used the assumption that the surface is locally spherical at each point. Under the same spherical assumption, Lee and Rosenfeld [131] computed the slant and tilt of the surface in the light source coordinate system using the first derivative of the intensity.

The approaches by Pentland and Tsai and Shah are linear approaches which linearize the reflectance map and solve for shape. Pentland [131] used the linear approximation of the reflectance function in terms of the surface gradient and applied a Fourier transform to the linear function to get a closed-form solution for the depth at each point. Tsai and Shah [179] applied the gradient’s discrete approximation first, then employed the linear approximation of the reflectance function in terms of the depth directly. Their algorithm recovered the depth at each point using a Jacobi iterative scheme.

Except for the traditional SFS work, we show in the following sections a two-level generative model for representing the images and surface depth maps of drapery and clothes. The upper level consists of a number of folds which will generate the high contrast (ridge) areas with a dictionary of shading primitives (for 2D images) and fold primitives (for 3D depth maps). These primitives are represented in parametric forms and are learned in a supervised learning phase using 3D surfaces of clothes acquired through photometric stereo. The lower level consists of the remaining flat areas which fill between the folds with a smoothness prior (Markov random field). We show that the classical ill-posed problem – shape from shading (SFS) can be much improved by this two-level model for its reduced dimensionality and incorporation of middle-level visual knowledge, i.e. the dictionary of primitives. Given an input image, we first infer the folds and compute a sketch graph using a sketch pursuit algorithm as in the primal sketch [81, 82]. The 3D folds are estimated by parameter fitting using the fold dictionary and they form the “skeleton” of the drapery/cloth surfaces. Then the lower level is computed by conventional SFS method using the fold areas as boundary



conditions. The two levels interact at the final stage by optimizing a joint Bayesian posterior probability on the depth map. We show a number of experiments which demonstrate more robust results in comparison with state-of-the-art work. In a broader scope, our representation can be viewed as a two-level inhomogeneous MRF model which is applicable to general shape-from-X problems. Our study is an attempt to revisit Marr’s idea [153] of computing the  $2\frac{1}{2}$ D sketch from primal sketch.

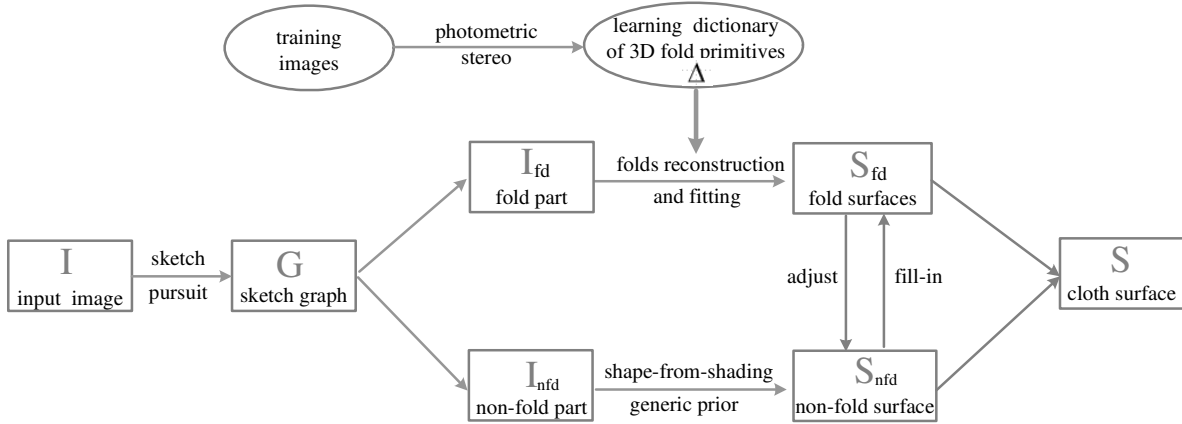


Figure 8.16: Reprinted with permission from [84]. The data-flow of our method for computing the 3D surface  $S$  of drapery/cloth from a single image  $I$  using the two-layer generative model. See text for interpretation.

### 8.3.1 Overview of the Two-Layer Generation Model

The dataflow of our method is illustrated in Figure 8.16 and a running example is shown in Figure 8.17. The problem is formulated in a Bayesian framework, and we adopt a stepwise greedy algorithm by minimizing various energy terms sequentially. Given an input image  $I$  on a lattice  $\Lambda$ , we first compute a sketch graph  $G$  for the folds by a greedy sketch pursuit algorithm. Figure 8.17.(b) is an exemplary graph  $G$ . The graph  $G$  has attributes for the shading and fold primitives.  $G$  decomposes the image domain into two disjoint parts: the fold part  $I_{fd}$  for pixels along the sketch and non-fold part  $I_{nfd}$  for the remaining flat areas. We estimate the 3D surface  $\hat{S}_{fd}$  for the fold part by fitting the 3D fold primitives in a fold dictionary  $\Delta_{fd}$ . Figure 8.17.(c) shows an example of  $S_{fd}$ . This will yield gradient maps  $(p_{fd}, q_{fd})$  for the fold surface. Then we compute the gradient maps  $(p_{nfd}, q_{nfd})$  for the non-fold part by the traditional shape-from-shading method on the lower level pixels, using gradient maps in the fold area as boundary conditions. Then we compute the joint surface  $S = (S_{fd}, S_{nfd})$  from the gradient maps  $(p, q)$  of both fold part and non-fold part. Therefore the computation of the upper level fold surfaces  $S_{fd}$  and the lower level flat surface  $S_{nfd}$  is coupled. Intuitively, the folds provide the global “skeleton” and therefore boundary conditions for non-fold areas, and the non-fold areas propagate information to infer the relative depth of the folds and to achieve a seamless surface  $S$ . The two-level generative model reduces to the traditional smoothness MRF model when the graph  $G$  is null. Since the two-layer generative model is similar to the one described in previous section, we skip the formulation part and show some qualitative results.

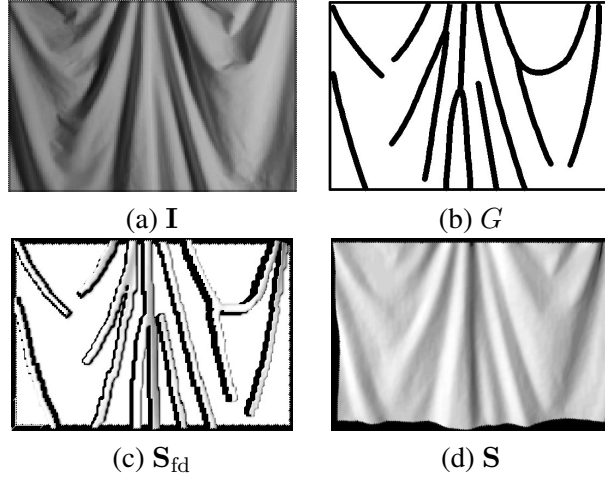


Figure 8.17: Reprinted with permission from [84]. (a). A drapery image under approximately parallel light. (b). The sketch graph for the computed folds. (c). The reconstructed surface for the folds. (d) The drapery surface after filling in the non-fold part. It is viewed at a slightly different angle and lighting direction.

### 8.3.2 Results

We test our whole algorithm on a number of images. Figure 8.18 shows the results for three images of drapery hung on wall and a cloth image (last column) on some people. The lighting direction and surface albedos for all the testing cloth are estimated by the method in [256].

In the experimental results, the first row are input images, the second row are the sketches of folds in the input images and their domain, the third row are the syntheses for  $I_{fd}$  based on the generative sketch model for the fold areas, the fourth row are the 3D reconstruction results  $S_{fd}$  for the fold areas, while fifth and sixth rows are the final reconstruction results of the whole cloth surface  $S$  shown in two novel views.

In these results, the folds in row (d) have captured most of the perceptually salient information in the input images, and they can reconstruct the surface without too much skewing effects. It makes sense to compute them before the non-fold part. We observe that the SFS for the non-fold parts indeed provides useful information for the 3D positions of the folds.

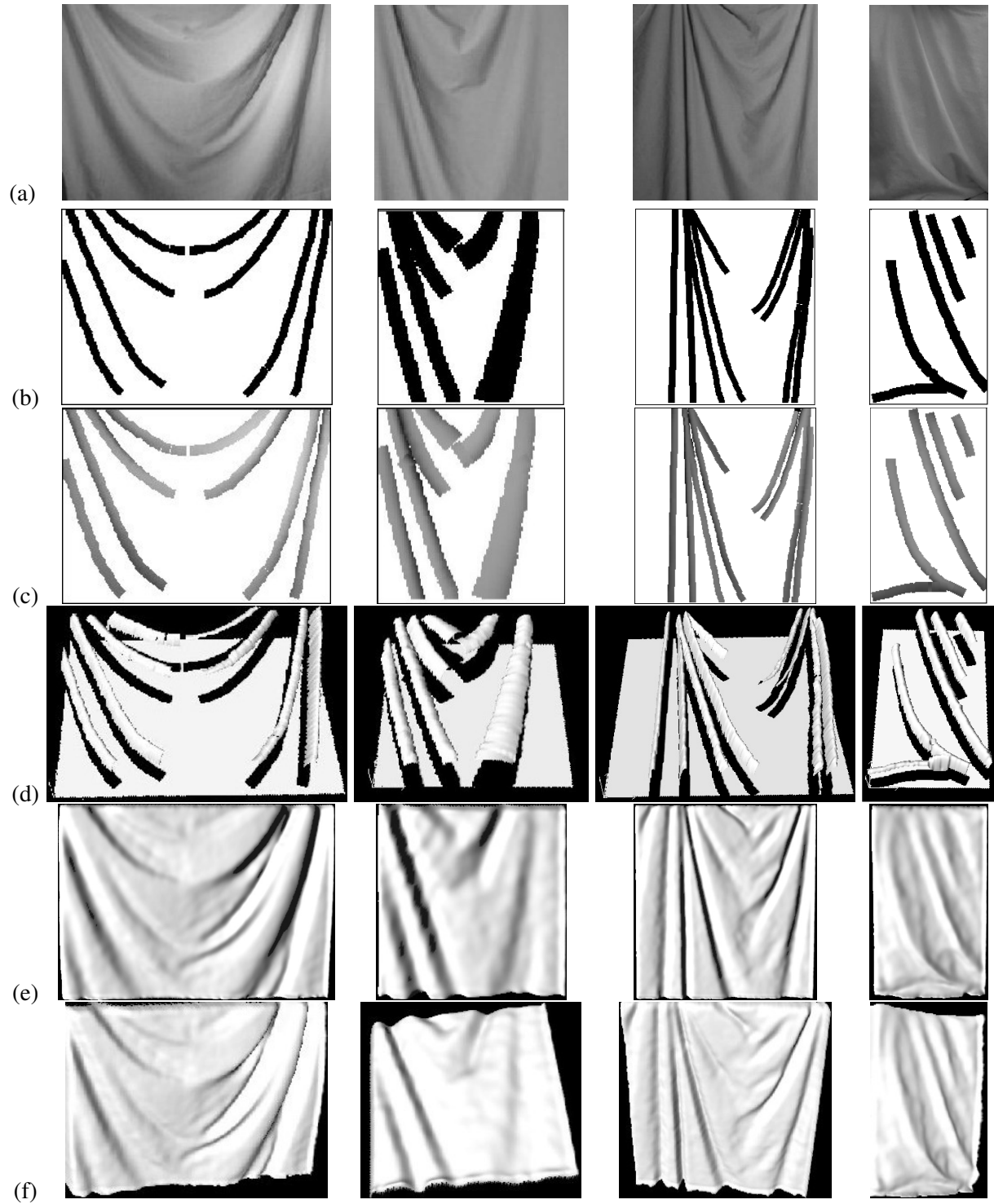


Figure 8.18: Reprinted with permission from [84]. (a). Input cloth image. (b). 2d folds and their image domains. (c). Synthesis for 2D fold sketches  $I_{fd}$ . (d). 3D reconstruction  $S_{fd}$  for fold areas. (e-f). Final reconstructed surface  $S$  in novel views.

## 8.4 2.5D Sketch from Direct Estimation

In previous two sections, we introduce two generative models that estimate 2.5D sketch from primal sketch. Recently, some discriminative methods that directly estimate the 2.5D sketch from images are developed with the help of large scale dataset and greatly improved computational ability. We review the recent large-scale dataset for the 2.5D sketch estimation, and some pioneer discriminative models for estimating the 2.5D sketch.

### 8.4.1 Dataset

#### Indoor Scene Dataset

Capturing the depth values directly for an image is hard without a proper sensing device. However, the development of Kinect provides Computer Vision researchers a feasible way to collect the data with depth value. Kinect is a line of motion sensing input devices that produced by Microsoft. The first-generation Kinect for Xbox 360 was introduced in November 2010 and the depth sensor in Kinect consists of an infrared laser projector combined with a monochrome CMOS sensor, which captures video data in 3D under any ambient light conditions. With the help of Kinect, several indoor scene understanding datasets have been introduced. Among them, NYU Depth dataset V2 [200] is the most widely used dataset in recent years. The dataset consists of 1449 RGBD images, gathered from a wide range of commercial and residential buildings in three different US cities, comprising 464 different indoor scenes across 26 scene classes. A dense per-pixel labeling was obtained for each image using Amazon Mechanical Turk.

#### Outdoor Scene Dataset

KITTI dataset [69] is a large-scale outdoor dataset for autonomous driving which provides various challenging benchmarks including stereo, optical flow, visual odometry / SLAM and 3D object detection. The benchmarks are captured by driving around a mid-size city, in rural areas and on highway. Their recording platform is equipped with two high resolution stereo camera systems (grayscale and color), a Velodyne HDL-64E laser scanner that produces more than one million 3D points per second and a localization system which combines GPS, GLONASS, an IMU and RTK correction signals. The cameras, laser scanner and localization system are calibrated and synchronized, providing accurate ground truth. KITTI provides benchmark for depth estimation, which makes it possible for direct depth estimation with data-driven learning methods.

### 8.4.2 Model

In this part, we introduce a representative discriminative approach [46] that uses convolutional neural network to estimate 2.5D sketch directly. This work presents a method that addresses the task of depth estimation by employing two deep network stacks: one that makes a coarse global prediction based on the entire image, and another that refines this prediction locally. It also applies a scale-invariant error to help measure depth relations rather than scale. The network is trained using a loss that explicitly accounts for depth relations between pixel locations, in addition to pointwise error.

The network is made of two component stacks, shown in Figure 8.19. A coarse-scale network first predicts the depth of the scene at a global level. This is then refined within local regions by a fine-scale network. Both stacks are applied to the original input, but in addition, the coarse network's output is passed to the fine network as additional first-layer image features. In this way, the local network can edit the global prediction to incorporate finer-scale details.

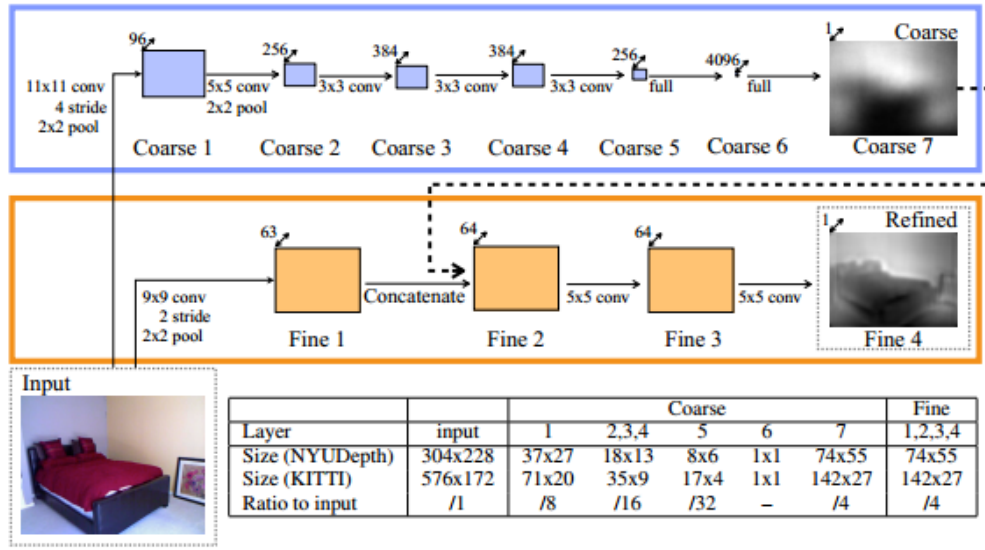


Figure 8.19: Reprinted with permission from [46]. Model architecture.

### A. Global coarse-scale network

The task of the coarse-scale network is to predict the overall depth map structure using a global view of the scene. The upper layers of this network are fully connected, and thus contain the entire image in their field of view. Similarly, the lower and middle layers are designed to combine information from different parts of the image through max-pooling operations to a small spatial dimension. In so doing, the network is able to integrate a global understanding of the full scene to predict the depth. Such an understanding is needed in the single-image case to make effective use of cues such as vanishing points, object locations, and room alignment. A local view (as is commonly used for stereo matching) is insufficient to notice important features such as these. As illustrated in Figure 8.19, the global, coarse-scale network contains five feature extraction layers of convolution and max-pooling, followed by two fully connected layers. The input, feature map and output sizes are also given in Figure 8.19. The final output is at 1/4-resolution compared to the input (which is itself downsampled from the original dataset by a factor of 2), and corresponds to a center crop containing most of the input (as we describe later, we lose a small border area due to the first layer of the fine-scale network and image transformations)

### B. Local fine-scale network

After taking a global perspective to predict the coarse depth map, we make local refinements using a second, fine-scale network. The task of this component is to edit the coarse prediction it receives to align with local details such as object and wall edges. The fine-scale network stack consists of convolutional layers only, along with one pooling stage for the first layer edge features. While the coarse network sees the entire scene, the field of view of an output unit in the fine network is 45x45 pixels of input. The convolutional layers are applied across feature maps at the target output size, allowing a relatively high-resolution output at 1/4 the input scale. More concretely, the coarse output is fed in as an additional low-level feature map. By design, the coarse prediction is the same spatial size as the output of the first fine-scale layer (after pooling), and we concatenate the two together (Fine 2 in Figure 8.19). Subsequent layers maintain this size using zero-padded convolutions.

### C. Scale-Invariant Error

The global scale of a scene is a fundamental ambiguity in depth prediction. Indeed, much of the error accrued using current elementwise metrics may be explained simply by how well the mean depth is predicted. However, using an oracle to substitute the mean log depth of each prediction with the mean from the corresponding ground truth reduces the error to 0.33, a 20% relative improvement. Likewise, for our system, these error rates are 0.28 and 0.22, respectively. Thus, just finding the average scale of the scene accounts for a large fraction of the total error. Motivated by this, we use a scale-invariant error to measure the relationships between points in the scene, irrespective of the absolute global scale. For a predicted depth map  $y$  and ground truth  $y^*$ , each with  $n$  pixels indexed by  $i$ , we define the scale-invariant mean squared error (in log space) as

$$D(y, y^*) = \frac{1}{2n} \sum_{i=1}^n (\log y_i - \log y_i^* + \alpha(y, y^*))^2 \quad (8.32)$$

where  $\alpha(y, y^*) = \frac{1}{n} \sum_i (\log y_i - \log y_i^*)$  is the value of  $\alpha$  that minimizes the error for a given  $(y, y^*)$ . For any prediction  $y$ ,  $e^\alpha$  is the scale that best aligns it to the ground truth. All scalar multiples of  $y$  have the same error, hence the scale invariance.

Two additional ways to view this metric are provided by the following equivalent forms. Setting  $d_i = \log y_i - \log y_i^*$  to be the difference between the prediction and ground truth at pixel  $i$ , we have

$$D(y, y^*) = \frac{1}{n} \sum_i d_i^2 - \frac{1}{n^2} \sum_{i,j} d_i d_j = \frac{1}{n} \sum_i d_i^2 - \frac{1}{n^2} \left( \sum_i d_i \right)^2 \quad (8.33)$$

### D. Training Loss

In addition to performance evaluation, we also tried using the scale-invariant error as a training loss. We set the per-sample training loss to

$$L(y, y^*) = \frac{1}{n} \sum_i d_i^2 - \frac{\lambda}{n^2} \left( \sum_i d_i \right)^2 \quad (8.34)$$

where  $d_i = \log y_i - \log y_i^*$  and  $\lambda \in [0, 1]$ . Note the output of the network is  $\log y$ ; that is, the final linear layer predicts the log depth. Setting  $\lambda = 0$  reduces to elementwise l2, while  $\lambda = 1$  is the scale-invariant error exactly. We use the average of these, *i.e.*  $\lambda = 0.5$ , finding that this produces good absolute-scale predictions while slightly improving qualitative output.

### E. Data Augmentation

We augment the training data with random online transformations:

- Scale: Input and target images are scaled by  $s \in [1, 1.5]$ , and the depths are divided by  $s$ .
- Rotation: Input and target are rotated by  $r \in [-5, 5]$  degrees.
- Translation: Input and target are randomly cropped to the sizes indicated in Figure 8.19.
- Color: Input values are multiplied globally by a random RGB value  $c \in [0.8, 1.2]^3$ .
- Flips: Input and target are horizontally flipped with 0.5 probability.

Note that image scaling and translation do not preserve the world-space geometry of the scene. This is easily corrected in the case of scaling by dividing the depth values by the scale  $s$  (making the image  $s$  times larger effectively moves the camera  $s$  times closer). Although translations are not easily fixed (they effectively change the camera to be incompatible with the depth values), we found that the extra data they provided benefited the network even though the scenes they represent were slightly warped. The other transforms, flips and in-plane rotation, are geometry-preserving.

### 8.4.3 Results

We show some qualitative results in Figure 8.20.

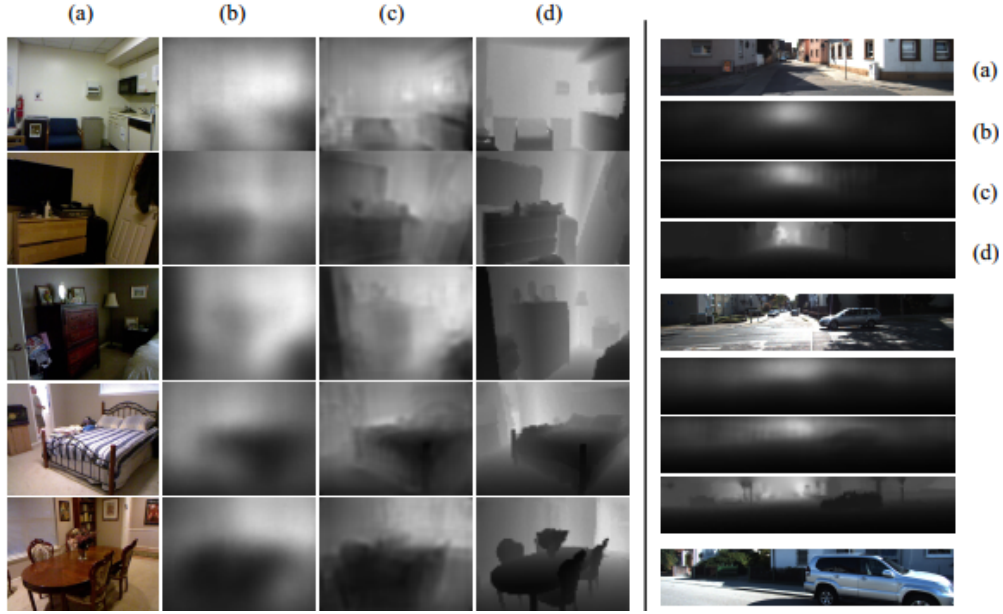


Figure 8.20: Reprinted with permission from [46]. Example predictions from our algorithm. NYUDepth [200] on left, KITTI [68] on right. For each image, we show (a) input, (b) output of coarse network, (c) refined output of fine network, (d) ground truth. The fine scale network edits the coarse-scale input to better align with details such as object boundaries and wall edges. Examples are sorted from best (top) to worst (bottom).





# 9

## Learning by Information Projection

In this chapter, a general framework for learning a statistical model as an approximation to the true distribution that generates images is considered.

### 9.1 Information Projection

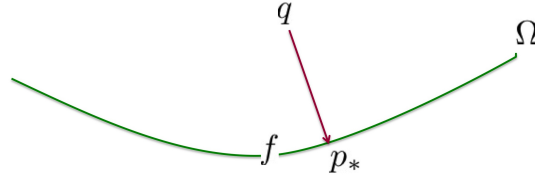


Figure 9.1: In the above illustration, each point is a probability distribution.  $f$  is the true distribution that generates the training examples.  $q$  is the reference distribution or the null model. The curve  $\Omega$  consists of all the distributions that reproduce the feature statistics of  $f$ .  $p_*$  is the projection of  $q$  onto  $\Omega$ .

Suppose training images  $\{\mathbf{I}_m, m = 1, \dots, M\} \sim f(\mathbf{I})$  are observed. The goal is to find a good approximation to the unknown true distribution  $f$  that generates the training examples. Suppose  $M$  is large so that it is feasible to estimate the expectations with respect to  $f$  accurately from the training examples.

Consider starting from a reference distribution  $q(\mathbf{I})$ , e.g., the white noise distribution. Suppose there is a set of features  $H(\mathbf{I}) = (H_k(\mathbf{I}), k = 1, \dots, K)$ . The following may be estimated:

$$\mathcal{E}_f[H(\mathbf{I})] \approx \frac{1}{M} \sum_{m=1}^M H(\mathbf{I}_m). \quad (9.1)$$

$\mathcal{E}_f[H(\mathbf{I})]$  is all that is known about the unknown  $f$  as far as the feature  $H$  is concerned.

Again, the goal is to find a distribution  $p$  to be an approximation to the unknown distribution  $f$ . Such a distribution should better reproduce the feature statistics that are most cared about, i.e.,

$$\mathcal{E}_p[H(\mathbf{I})] = \mathcal{E}_f[H(\mathbf{I})]. \quad (9.2)$$

Call  $p$  such an eligible distribution. Let

$$\Omega = \{p : \mathcal{E}_p[H(\mathbf{I})] = \mathcal{E}_f[H(\mathbf{I})]\} \quad (9.3)$$

be the family of all the eligible distributions. Clearly,  $f \in \Omega$ . See Fig. 9.1 for an illustration, in which each point is a probability distribution.  $f$  is the true distribution that generates the training examples.  $q$  is the reference distribution or the null model. The curve  $\Omega$  consists of all the distributions that reproduce the feature statistics of  $f$ .

So starting from  $q$ , the updated goal is to find a distribution in  $\Omega$  so that it has the minimum distance from  $q$ , i.e., an eligible distribution that can be obtained from a minimum modification of  $q$ , so that artificial features beyond  $H$  are not introduced. Let  $p_*$  be such a distribution.  $p_*$  may be thought of as the projection of  $q$  onto the family  $\Omega$ , and it is hence called the information projection. More specifically, the objective is to find

$$p_* = \arg \min_{p \in \Omega} \text{KL}(p||q). \quad (9.4)$$

### 9.1.1 Orthogonality and Duality

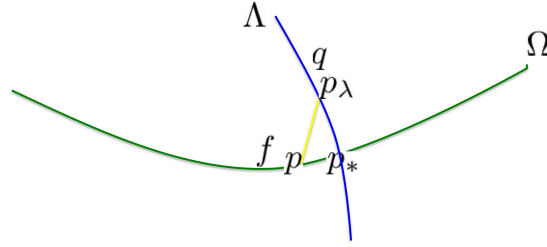


Figure 9.2: The eligible family  $\Omega$  and the model family  $\Lambda$  are orthogonal,  $\Omega \perp \Lambda$ , because for any  $p \in \Omega$ ,  $p_\lambda \in \Lambda$ ,  $\text{KL}(p||p_\lambda) = \text{KL}(p||p_*) + \text{KL}(p_*||p_\lambda)$ , in which  $p_* = \Omega \cap \Lambda$  is the intersection.

In order to solve for  $p_* = \arg \min_{p \in \Omega} \text{KL}(p||q)$ , the Langevin multiplier may be used as it was for the FRAME model. In this chapter, an approach is adopted that is less direct but is more geometrically meaningful. A dual minimization problem may be found by introducing another family of distributions that is called the model family. Specifically, the following exponential family models are defined:

$$p(\mathbf{I}; \lambda) = \frac{1}{Z(\lambda)} \exp [\langle \lambda, H(\mathbf{I}) \rangle] q(\mathbf{I}), \quad (9.5)$$

in which  $\lambda = (\lambda_k, k = 1, \dots, K)$ ,  $\langle \lambda, H(\mathbf{I}) \rangle = \sum_{k=1}^K \lambda_k H_k(\mathbf{I})$ , and

$$Z(\lambda) = \int \exp [\langle \lambda, H(\mathbf{I}) \rangle] q(\mathbf{I}) = \mathcal{E}_q[\exp [\langle \lambda, H(\mathbf{I}) \rangle]] \quad (9.6)$$

is the normalizing constant. For simplicity,  $p(\mathbf{I}; \lambda)$  as  $p_\lambda$  is written. Let

$$\Lambda = \{p_\lambda, \forall \lambda\} \quad (9.7)$$

be the model family. Clearly,  $q \in \Lambda$  with  $\lambda = 0$ . See Fig. 9.2 for an illustration, in which the model family  $\Lambda$  is illustrated by the vertical curve.

Let  $p_* = p(\mathbf{I}; \lambda_*) = \Omega \cap \Lambda$  be the intersection between the eligible family and the model family. It shall be shown that  $p_* = \arg \min_{p \in \Omega} \text{KL}(p||q)$ , which is the projection that is sought after. The key is that  $\Omega \perp \Lambda$ ; that is,  $\Omega$  is orthogonal to  $\Lambda$ , in the sense of the following Pythagorean theorem [35]:

**Theorem 1.** For any  $p_\lambda \in \Lambda$  and any  $p \in \Omega$ :

$$\text{KL}(p||p_\lambda) = \text{KL}(p||p_\star) + \text{KL}(p_\star||p_\lambda). \quad (9.8)$$

Proof:

$$\text{KL}(p||p_\lambda) = \mathcal{E}_p[\log p(\mathbf{I})] - \mathcal{E}_p[\log p(\mathbf{I}; \lambda)]. \quad (9.9)$$

$$\text{KL}(p||p_\star) = \mathcal{E}_p[\log p(\mathbf{I})] - \mathcal{E}_p[\log p(\mathbf{I}; \lambda_\star)]. \quad (9.10)$$

$$\text{KL}(p_\star||p_\lambda) = \mathcal{E}_{p_\star}[\log p(\mathbf{I}; \lambda_\star)] - \mathcal{E}_{p_\star}[\log p(\mathbf{I}; \lambda)]. \quad (9.11)$$

Meanwhile,  $\mathcal{E}_p[\log p(\mathbf{I}; \lambda_\star)] = \mathcal{E}_{p_\star}[\log p(\mathbf{I}; \lambda_\star)]$ , and  $\mathcal{E}_p[\log p(\mathbf{I}; \lambda)] = \mathcal{E}_{p_\star}[\log p(\mathbf{I}; \lambda)]$ , because  $\mathcal{E}_p[H(\mathbf{I})] = \mathcal{E}_{p_\star}[H(\mathbf{I})]$ , as both  $p$  and  $p_\star$  belong to  $\Omega$ . Thus, the result follows.  $\square$

The above result leads to the following duality result:

$$p_\star = \arg \min_{p \in \Omega} \text{KL}(p||q) = \arg \min_{p_\lambda \in \Lambda} \text{KL}(f||p_\lambda). \quad (9.12)$$

Thus, it is seen that  $p_\star = \arg \min_{p \in \Omega} \text{KL}(p||q)$  by finding  $p_\star = \arg \min_{p_\lambda \in \Lambda} \text{KL}(f||p_\lambda)$ . Since  $\text{KL}(f||p_\lambda) = \mathcal{E}_f[\log f(\mathbf{I})] - \mathcal{E}_f[\log p(\mathbf{I}; \lambda)]$ ,

$$p_\star = \arg \min_{p_\lambda \in \Lambda} \text{KL}(f||p_\lambda) = \arg \max_{p_\lambda \in \Lambda} \mathcal{E}_f[\log p(\mathbf{I}; \lambda)]. \quad (9.13)$$

$\mathcal{E}_f[\log p(\mathbf{I}; \lambda)]$  is actually the log-likelihood in the limit.

### 9.1.2 Maximum Likelihood Implementation

If  $\{\mathbf{I}_m, m = 1, \dots, M\} \sim f(\mathbf{I})$  is observed, then

$$\mathcal{E}_f[\log p(\mathbf{I}; \lambda)] \approx \frac{1}{M} \sum_{m=1}^M \log p(\mathbf{I}_m; \lambda), \quad (9.14)$$

so  $\lambda_\star$  can be approximated by the maximum likelihood estimate  $\hat{\lambda} = \arg \max_{\lambda} L(\lambda)$ , in which

$$L(\lambda) = \frac{1}{M} \sum_{m=1}^M \log p(\mathbf{I}_m; \lambda) = \frac{1}{M} \sum_{m=1}^M \langle \lambda, H(\mathbf{I}_m) \rangle - \log Z(\lambda) \quad (9.15)$$

is the log-likelihood function of the exponential family model (9.5).

It can be shown that

$$\frac{\partial}{\partial \lambda} \log Z(\lambda) = \mathcal{E}_\lambda[H(\mathbf{I})], \quad (9.16)$$

in which  $\mathcal{E}_\lambda$  denotes the expectation with respect to  $p(\mathbf{I}; \lambda)$ .

$$\frac{\partial^2}{\partial \lambda^2} \log Z(\lambda) = \text{Var}_\lambda[H(\mathbf{I})], \quad (9.17)$$

in which  $\text{Var}_\lambda$  denotes the variance with respect to  $p(\mathbf{I}; \lambda)$ . Thus,

$$\frac{\partial}{\partial \lambda} L(\lambda) = \frac{1}{M} \sum_{m=1}^M H(\mathbf{I}_m) - \mathcal{E}_\lambda[H(\mathbf{I})], \quad (9.18)$$

and

$$\frac{\partial^2}{\partial \lambda^2} L(\lambda) = \text{Var}_\lambda[H(\mathbf{I})]. \quad (9.19)$$

That is,  $L(\lambda)$  is a concave function with a unique maximum, provided that  $\text{Var}[H(\mathbf{I})]$  is positive definite, which is the case if the components of  $H(\mathbf{I})$  are linearly independent. At the maximum  $\hat{\lambda}$ ,

$$\mathcal{E}_{\hat{\lambda}}[H(\mathbf{I})] = \frac{1}{M} \sum_{m=1}^M H(\mathbf{I}_m), \quad (9.20)$$

in which  $\mathcal{E}_\lambda$  denotes the expectation with respect to  $p(\mathbf{I}; \lambda)$ . Thus, at the maximum likelihood estimate, the model reproduces the observed feature statistics.

If  $M \rightarrow \infty$ ,  $L(\lambda) \rightarrow \mathcal{E}_f[\log p(\mathbf{I}; \lambda)]$ , and at the maximum,  $\mathcal{E}_f[H(\mathbf{I})] = \mathcal{E}_{\lambda_*}[H(\mathbf{I})]$ , i.e.,  $p_{\hat{\lambda}} \rightarrow p_* = \Lambda \cap \Omega$ .

Crucially, the information projection viewpoint is deeper than the maximum likelihood estimation of the exponential family model (9.5). The former provides a justification for the latter, and the latter is to implement the former.

### 9.1.3 The Minimax Learning Framework

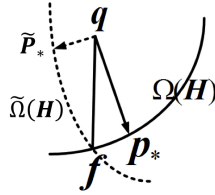


Figure 9.3: The solid curve and the dotted curve illustrate two eligible families defined by two different sets of feature statistics.  $q$  should be projected onto the solid curve instead of the dotted curve in order to get closer to the target distribution  $f$ .

Suppose there are two different sets of features  $H(\mathbf{I})$  and  $\tilde{H}(\mathbf{I})$ ; then, there are two different eligible families  $\Omega$  and  $\tilde{\Omega}$ . If the same reference distribution  $q$  is projected onto  $\Omega$  and  $\tilde{\Omega}$  respectively,  $p_*$  and  $\tilde{p}_*$  will be the results, respectively. In Fig. 9.3, the solid curve illustrates  $\Omega$ , while the dotted curve illustrates  $\tilde{\Omega}$ . Due to the Pythagorean theorem,

$$\text{KL}(f||q) = \text{KL}(f||p_*) + \text{KL}(p_*||q) = \text{KL}(f||\tilde{p}_*) + \text{KL}(\tilde{p}_*||q). \quad (9.21)$$

Thus, if the desire is to make  $\text{KL}(f||p_*)$  small,  $\text{KL}(p_*||q)$  needs to be made large. So if there are many different choices of  $H$ , then the one that maximizes  $\text{KL}(p_*||q)$  should be chosen. Recall that  $p_* = \arg \min_{p \in \Omega} \text{KL}(p||q)$ ; thus, the goal is to solve the following max-min problem:

$$\max_H \min_{p \in \Omega(H)} \text{KL}(p||q), \quad (9.22)$$

in which  $\Omega(H)$  is the eligible family defined by the set of features  $H$ . Because of the duality, the above problem is equivalent to the maximum likelihood problem,

$$\max_H \min_\lambda \text{KL}(f||p_{H,\lambda}), \quad (9.23)$$

in which  $p_{H,\lambda}$  is the exponential family model defined by  $H$  in equation (9.5). Here  $H$  is made explicit in  $p_{H,\lambda}$  because different sets of features are being considered. Thus, the log-likelihood  $L$  in (9.15) can be maximized over both  $\lambda$  and  $H$ .

Intuitively, for a given set of features  $H$  that defines an eligible family  $\Omega(H)$ , the goal is to choose  $p$  that is closest to  $q$  to avoid adding artificial features that are not in  $H$ . Meanwhile, for different sets of features, the set of features so that the change from  $q$  to the corresponding  $p_*$  is the largest should be chosen.

The minimax entropy learning [262] is a special case of the above learning scheme, in which the reference distribution  $q$  is the uniform measure.

### 9.1.4 Model Pursuit Strategies

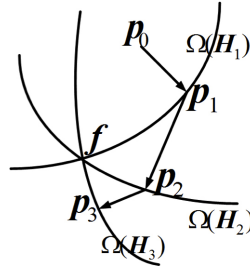


Figure 9.4: Learning a sequence of distributions  $p_k$  to approach the target distribution  $f$ . Each time, the current distribution  $p_{k-1}$  is projected onto the eligible family defined by  $H_k$  to obtain  $p_k$ .

$H(\mathbf{I}) = (H_k(\mathbf{I}), k = 1, \dots, K)$  may be obtained by selecting each  $H_k$  sequentially to pursue a sequence of models  $p_k$  that get closer and closer to the target distribution  $f$ . Fig. 9.4 illustrates the idea of sequential projection. The starting point is  $p_0 = q$ , the reference distribution or the null model. After selecting the first feature  $H_1$ , the eligible family  $\Omega(H_1)$  that consists of all the distributions that reproduce  $\mathcal{E}_f[H_1(\mathbf{I})]$  is obtained. Then,  $p_0$  is projected onto  $\Omega(H_1)$  to obtain  $p_1$ . Then, the second feature  $H_2$  is selected and projected  $p_1$  onto  $\Omega(H_2)$  to obtain  $p_2$ , and so forth. Because of the Pythagorean theorem, each iteration approaches the target  $f$ .

Sequential projection leads to the following greedy strategy to choose  $H_k$  sequentially. At each step, the maximum reduction in the distance from the current model to the target distribution is sought. Specifically, let  $p_{k-1}$  be the current model.  $H_k = \arg \max \text{KL}(p_k || p_{k-1})$  is chosen, which can be implemented by the maximum likelihood of the following exponential family model:

$$p_k(\mathbf{I}) = \frac{1}{Z_k(\lambda_k)} \exp [\lambda_k H_k(\mathbf{I})] p_{k-1}(\mathbf{I}), \quad (9.24)$$

in which  $p_{k-1}$  plays the role of the current reference distribution, and both  $H_k$  and  $\lambda_k$  are obtained by maximizing the likelihood function of (9.24) as a function of  $H_k$  and  $\lambda_k$ . In the end, a model of the form (9.5) is obtained.

In the above discussion, it is assumed that there is a large dictionary of features  $\{H_i, i = 1, \dots, N\}$ , and a small number of them from this large dictionary may be selected. A related strategy for feature selection is via  $\ell_1$  regularization, as in basis pursuit [25] or Lasso [210]. Specifically, the following full model is assumed, instead of the final selected model:

$$p(\mathbf{I}; \lambda) = \frac{1}{Z(\lambda)} \exp \left[ \sum_{i=1}^N \lambda_i H_i(\mathbf{I}) \right] q(\mathbf{I}), \quad (9.25)$$

in which  $\lambda = (\lambda_i, i = 1, \dots, N)$  is a long vector. The vector  $\lambda$  is assumed to be sparse, i.e., only a small number of its components is different from zero. Let  $L(\lambda)$  be the log-likelihood of the above full model; model selection can be performed by maximizing the  $\ell_1$ -regularized log-likelihood,  $L(\lambda) + \rho|\lambda|$ , in which  $|\lambda| = \sum_{i=1}^N |\lambda_i|$  is the  $\ell_1$  norm of  $\lambda$ , and  $\rho$  is a tuning constant. The maximization of the penalized log-likelihood  $L(\lambda) + \rho|\lambda|$  can be accomplished by an epsilon-boosting algorithm [62, 187], in which at each step, the component of  $L'(\lambda)$  is chosen that has the maximum magnitude, and then this component is updated by a small amount  $\epsilon$ .

It is also possible that the dictionary of the features are parametrized by some continuous parameters  $\gamma$ , so the model is

$$p(\mathbf{I}; \theta) = \frac{1}{Z(\theta)} \exp \left[ \sum_{i=1}^N \lambda_i H_i(\mathbf{I}; \gamma) \right] q(\mathbf{I}), \quad (9.26)$$

in which  $\theta = (\lambda, \gamma)$ . Both  $\lambda$  and  $\gamma$  may be learned by maximum likelihood.

## 9.2 A Unifying View

### 9.2.1 Relation to Discriminative Learning

Suppose negative examples from the reference distribution  $q(\mathbf{I})$  are observed, and positive examples from the model  $p(\mathbf{I}; \lambda)$  in (9.5) are observed. Let  $\alpha$  be the prior probability that a positive example is observed. Then the posterior probability that an example  $\mathbf{I}$  is a positive example is

$$p(+|\mathbf{I}) = \frac{1}{1 + \exp \left[ - \sum_{k=1}^K \lambda_k H_k(\mathbf{I}) - b \right]}, \quad (9.27)$$

in which  $b = \log[\alpha/(1 - \alpha)]$ . This is a logistic regression model. If examples from multiple categories are observed, a multinomial logistic regression will be obtained.

The learning method in the previous section can be considered a generative version of adaboost [54].

### 9.2.2 Learning FRAME

So far, not much detail has been given about the features. In this section, the idea of information projection using concrete examples of learning two-dimensional distributions, in which the feature statistics are linear projections or filter responses, shall be elaborated.

Fig. 9.5 illustrates two examples of information projection. The training examples  $\{\mathbf{I}_m, m = 1, \dots, M\}$  are two-dimensional, i.e., they are images of different two dimensional points. The scatterplot of the data forms a two-dimensional cloud of points. The features are of the form  $H_k(\mathbf{I}) = h(\langle \mathbf{I}, B_k \rangle)$ , in which  $B_k$  is also a two-dimensional vector, just like  $\mathbf{I}$ .  $\langle \mathbf{I}, B_k \rangle$  is the projection of  $\mathbf{I}$  on  $B_k$ . One may also call it a filter response, in which  $B_k$  plays the role of a filter.  $h(r)$  is a one-hot indicator vector. Specifically, the range of  $\langle \mathbf{I}, B_k \rangle$  is divided into a finite number of  $L$  bins, so that  $h(r) = (h_l(r), l = 1, \dots, L)$ .  $h_l(r) = 1$  if  $r$  falls into the  $l$ -th bin, and  $h_l(r) = 0$  otherwise. Thus,  $\sum_{m=1}^M H_k(\mathbf{I}_m)/M = \sum_{m=1}^M h(\langle \mathbf{I}_m, B_k \rangle)/M$  is the histogram of the projected points  $\{\langle \mathbf{I}_m, B_k \rangle, m = 1, \dots, M\}$  projected onto  $B_k$ . It may be assumed that the squared length  $|B_k|^2 = 1$  and that the direction  $B_k$  may be discretized in the two-dimensional domain.

Consider starting from the uniform distribution over the two-dimensional domain of  $\mathbf{I}$ , assumed to be the unit square. Then, apply the model pursuit strategy by selecting  $B_k, k = 1, \dots, K$ . Each time, a  $B_k$  is

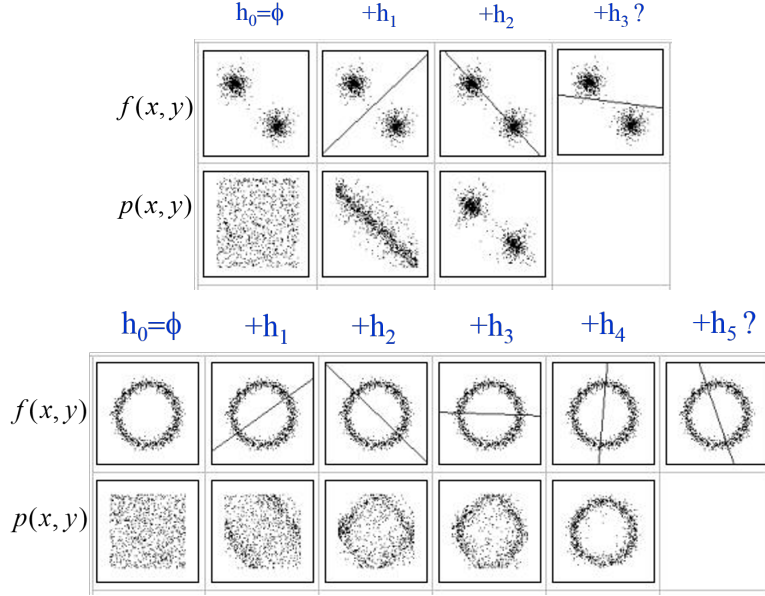


Figure 9.5: Reprinted with permission from [138]. This is an example of learning two dimensional distributions by information projection. Each step, the marginal distribution of the data points projected onto a selected vector is matched.

selected, and the marginal histogram of the projected points is matched. After a number of steps, a model

$$p(\mathbf{I}; \mathbf{B}, \lambda) = \frac{1}{Z} \exp \left[ \sum_{k=1}^K \lambda_k h(\langle \mathbf{I}, B_k \rangle) \right] q(\mathbf{I}) \quad (9.28)$$

is pursued, in which  $\lambda = (\lambda_k, k = 1, \dots, K)$  and  $\mathbf{B} = (B_k, k = 1, \dots, K)$  is the learned dictionary of projections or filters.

Fig. 9.5 illustrates the learning process. In each example, the first row displays the target distribution  $f$ , as well as the selected direction or filter  $B_k$  for each  $k$ . The second row displays the learned model  $p$  as more directions are added. After adding only a small number of filters, the learned model  $p$  is very similar to  $f$ . The learning method is related to projection pursuit [60].

In Fig. 9.5, the starting point is the uniform distribution. One can also start from a Gaussian white noise model with a small variance.

In addition to the pursuit strategy, the dictionary  $\mathbf{B}$  may also be learned directly by maximum likelihood, by taking derivatives with respect to both  $\lambda$  and  $\mathbf{B}$ . In order to take the derivative with respect to  $B_k$ ,  $h$  needs to be made continuous and differentiable. A possible choice is the rectified linear unit  $h(r) = \max(0, r - b)$ , in which  $b$  is the threshold, which can also be estimated by maximum likelihood.

Although the two toy examples are simple, they are very illustrative and have deep implications. The model (9.28) can be extended to model large images by making the filters  $B_k$  convolutional, i.e.,  $B_k$  is a localized image patch (e.g.,  $7 \times 7$ ) and  $B_k$  is applied around each pixel. If this model is learned from natural images, Gabor filters and Differences of Gaussian filters will be learned. Model (9.28) is the simplest FRAME model.

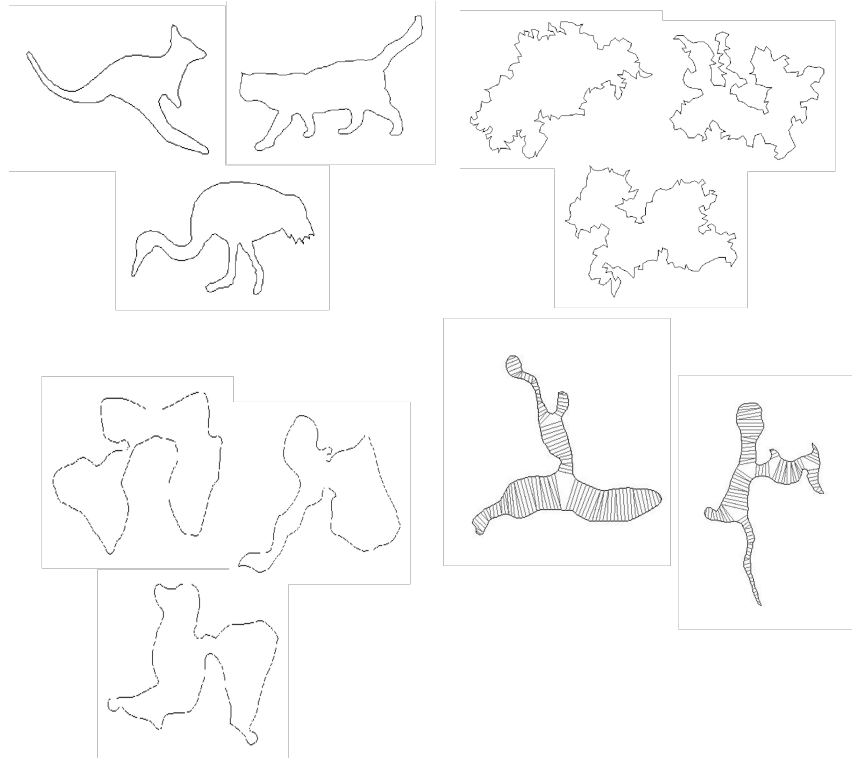


Figure 9.6: Reprinted with permission from [257]. This is an example of learning a sequence of models for shapes by adding shape statistics.

### 9.2.3 Learning Shape Patterns

In addition to learning image appearance patterns, shape patterns may also be learned by information projection. Fig. 9.6 illustrates an example of learning generic object shapes by adding shape statistics. Fig. 9.7 illustrates an example of learning specific face shapes by adding relevant statistics.



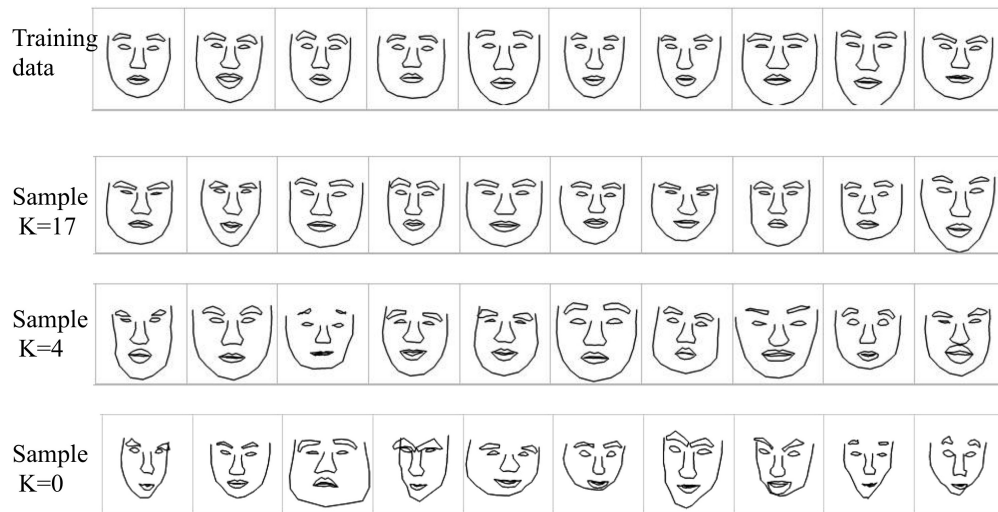


Figure 9.7: Reprinted with permission from [138]. This is an example of learning a sequence of models for face shapes by adding shape statistics.



# 10

## Information Scaling and Regimes of Models

One fundamental property of natural image data that distinguishes vision from other sensory tasks such as speech recognition is that scale plays a profound role in image formation and interpretation. Specifically, visual objects can appear at a wide range of scales in the images due to the change of viewing distance as well as camera resolution. The same objects appearing at different scales produce different image data with different statistical properties. Fig. (10.1) shows two examples of information scaling, where the change of scale causes the change of image properties, which may trigger the change of the modeling scheme for image representation.



Figure 10.1: Images of the same objects can appear very different at different viewing distances or camera resolution, a phenomenon we call information scaling.

In this section, we study the change of statistical properties, in particular, some information theoretical properties, of the image data over scale. We show that the *entropy rate*, defined as entropy per pixel, of the image data changes over scale. Moreover, the inferential uncertainty of the outside scene that generates the image data also changes with scale. We call these changes information scaling.

### 10.1 Image Scaling

To give the reader some concrete ideas, we first study information scaling empirically by experimenting with the so-called dead leaves model.

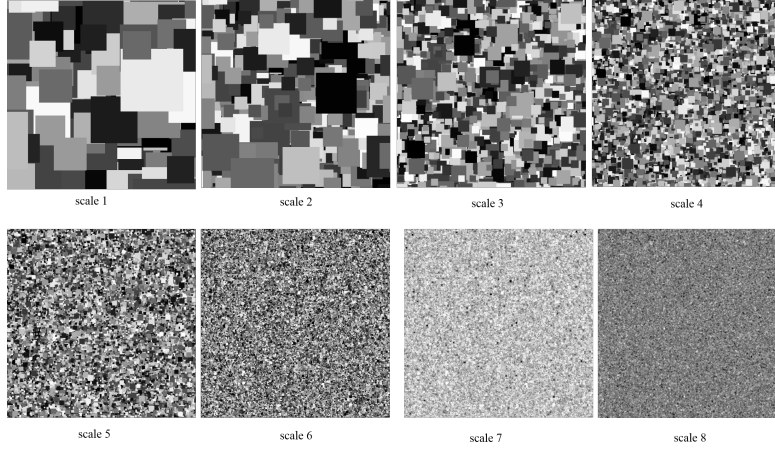


Figure 10.2: Reprinted with permission from [235]. Pictures of the simulated ivy wall taken at 8 viewing distances. The viewing distance of the  $i + 1$ -st image is twice that of the  $i$ -th image.

## Model and assumptions

The dead leaves model [155] was used by Lee, et al. [129] in their investigation in image statistics of natural scenes, which was also previously used to model natural images. For our purpose, we may consider that the model describes an ivy wall covered by a large number of leaves of similar sizes. See Fig. (10.2) for some examples. We assume that the leaves are of squared shape and are uniformly colored. Each leaf is represented by:

1. Its length or width  $r$ , which follows a distribution  $f(r) \propto 1/r^3$  over a finite range  $[r_{\min}, r_{\max}]$ .
2. Its color or shade  $a$ , which follows a uniform distribution over  $[a_{\min}, a_{\max}]$ .
3. Its position  $(x, y, z)$ , where the wall serves as the  $(x, y)$  plane, and  $z \in [0, z_{\max}]$  is the distance between the leaf and the wall. We assume that  $z_{\max}$  is very small, so that  $z$  matters only for deciding the occlusions among the leaves.

For the collection of leaves  $\{(r_k, a_k, x_k, y_k, z_k)\}$ , we assume that  $r_k$  are independent of each other, and so are  $a_k$ .  $(x_k, y_k, z_k)$  follows a Poisson process in  $\mathbf{R}^2 \times [0, z_{\max}]$ . We assume that the intensity of the Poisson process  $\lambda$  is large enough so that the leaves completely cover the wall. As noted by Lee et al. (2001),  $\{(r_k, a_k, x_k, y_k, z_k)\}$  is a Poisson process in the joint domain  $[r_{\min}, r_{\max}] \times [a_{\min}, a_{\max}] \times \mathbf{R}^2 \times [0, z_{\max}]$  with respect to the measure  $f(r)drda\lambda dx dy dz$ .

Lee et al. (2001) shows that this Poisson process is scale invariant under the assumption that  $[r_{\min}, r_{\max}] \rightarrow [0, \infty]$ . Specifically, under the scaling transformation  $x' = x/s$  and  $y' = y/s$ , where  $s$  is a scaling parameter, we have  $r' = r/s$ , and the Poisson process will be distributed in  $[r_{\min}/s, r_{\max}/s] \times [a_{\min}, a_{\max}] \times \mathbf{R}^2 \times [0, z_{\max}]$  with respect to the measure  $f(sr')sdr'da\lambda sdx'sdy'dz$ , which is equal to  $f(r')dr'da\lambda dx'dy'dz'$  because  $f(r) \propto 1/r^3$ . As  $[r_{\min}, r_{\max}] \rightarrow [0, \infty]$ ,  $[r_{\min}/s, r_{\max}/s] \rightarrow [0, \infty]$  too, so the Poisson process is invariant under the scaling transformation. The assumption of Lee et al. (2000) appears to hold for most of the studies of natural image statistics.

However, in our experiment,  $[r_{\min}, r_{\max}]$  is assumed to be a relatively narrow range. Under the scaling transformation, this range will change to  $[r_{\min}/s, r_{\max}/s]$ , which is far from being invariant. From this perspective, we may consider that Lee et al. (2001) and the papers cited above are concerned with the

marginal statistics by integrating over the whole range of scale. Our work, however, is concerned with the conditional statistics given a narrow range of scale, especially how such conditional statistics change under the scaling transformation. While it is important to look at the marginal statistics over the whole range of scale, it is perhaps even more important to study the conditional statistics at different scales in order to model different image patterns. Moreover, the conditional statistics at different scales may have to be accounted for by different regimes of statistical models.

## Image formation and scaling

Let  $O_k \subset \mathbf{R}^2$  be the squared area covered by leaf  $k$  in the  $(x, y)$  domain of the ivy wall. Then the scene of the ivy wall can be represented by a function  $W(x, y) = a_{k(x, y)}$ , where  $k(x, y) = \arg \max_{k: (x, y) \in O_k} z_k$ , i.e., the most forefront leaf that covers  $(x, y)$ .  $W(x, y)$  is a piecewise constant function defined on  $\mathbf{R}^2$ .

Now let's see what happens if we take a picture of  $W(x, y)$  from a distance  $d$ . Suppose the scope of the domain covered by the camera is  $\Omega \subset \mathbf{R}^2$ , where  $\Omega$  is a finite rectangular region. As noted by Mumford and Gidas (2001), a camera or a human eye only has a finite array of sensors or photoreceptors. Each sensor receives light from a small neighborhood of  $\Omega$ . As a simple model of the image formation process, we may divide the continuous domain  $\Omega$  into a rectangular array of squared windows of length or width  $\sigma d$ , where  $\sigma$  is decided by the resolution of the camera. Let  $\{\Omega_{ij}\}$  be these squared windows, with  $(i, j) \in D$ , where  $D$  is a rectangular lattice.

The image  $\mathbf{I}$  is defined on  $\Lambda$ . Let  $s = d\sigma$  be the scale parameter of the image formation process, then

$$\mathbf{I}_s(i, j) = \frac{1}{s^2} \int_{\Omega_{ij}} W(x, y) dx dy, \quad (i, j) \in D, \quad (10.1)$$

which is the average of  $W(x, y)$  within window  $\Omega_{ij}$ . Equation (10.1) can also be written as

$$w_s(x, y) = \frac{1}{s^2} \int W(x', y') g((x - x')/s, (y - y')/s) dx' dy' = W * g_s; \quad (10.2)$$

$$\mathbf{I}_s(i, j) = w_s(u + is, v + js), \quad (10.3)$$

where  $g$  is a uniform density function within the window  $[-1/2, 1/2] \times [-1/2, 1/2]$ , and  $g_s(x, y) = g(x/s, y/s)/s^2$ .  $(u, v) \in [0, s]^2$  denotes the small shifting of the rectangular lattice. There are two operations involved. Equation (10.2) is smoothing:  $w_s$  is a smoothed version of  $W$ . Equation (10.3) is subsampling:  $\mathbf{I}_s$  is a discrete sampling of  $w_s$ . To be more general,  $g$  in Equation (10.2) can be any density function, for instance, Gaussian density function.

The scale parameter  $s$  can be changed by either changing the viewing distance  $d$  or the camera resolution  $\sigma$ . If we increase  $s$  by increasing the viewing distance or zooming out the camera, then both the size of the scope  $\Omega$  and the size of the windows  $\Omega_{ij}$  will increase proportionally. So the resulting image  $\mathbf{I}_s$  will change. For example, if we double  $s$  to  $2s$ , then  $\mathbf{I}_{2s}$  will cover a scope 4 times as large as the scope of  $\mathbf{I}_s$ . Because each squared window of size  $2s$  contains 4 squared windows of size  $s$ , if we look within the portion of  $\mathbf{I}_{2s}$  that corresponds to  $\mathbf{I}_s$ , then the intensity of a pixel in  $\mathbf{I}_{2s}$  is the block average of the intensities of the corresponding  $2 \times 2$  pixels in  $\mathbf{I}_s$ .

If  $g$  is a Gaussian kernel, then the set of  $\{w_s(x, y), s > 0\}$  forms a scale space. The scale space theory can account for the change of image intensities due to scaling. But it does not explain the change of statistical properties of the image data under the scaling transformation.

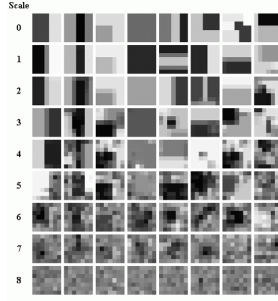


Figure 10.3: Reprinted with permission from [235]. The  $7 \times 7$  local patches taken from the images at different scales.

### Empirical observations on information scaling

Figure (10.2) shows a sequence of 8 images of  $W$  taken at 8 viewing distances. The images are generated according to Equation (10.1). The viewing distance of the  $i + 1$ -st image is twice that of the  $i$ -th image. So the viewing distance of the last image is 128 times that of the first image. Within this wide range of viewing distance, the images display markedly different statistical properties even though they are generated by the same  $W$ . The reason is that the square leaves appear at different scales in different images.

(1) For an image taken at near distance, such as image (1), the window size of a pixel is much less than the average size of the leaves, i.e.,  $s \ll r$ . The image can be represented deterministically by a relatively small number of occluding squares, or by local geometric structures such as edges, corners, etc. The constituent elements of the image are squares or local geometrical structures, instead of pixels.

(2) For an image at intermediate distance, the window size of a pixel becomes comparable to the average size of leaves, i.e.,  $s \approx r$ . The image becomes more complex. For images (4) and (5), they cannot be represented by a small number of geometrical structures anymore. The basic elements have to be pixels themselves. If a simple interpretation of the image is sought after, this interpretation has to be some sort of simple summary that cannot code the image intensities deterministically. The summary can be in the form of some spatial statistics of image intensities.

(3) For an image at a far distance, the window size of a pixel can be much larger than the average size of the squares, i.e.,  $s \gg r$ . Each pixel covers a large number of leaves, and its intensity value is the average of many leaves. The image is approaching the white noise.

Computer vision algorithms always start from the analysis of local image patches, often at multiple resolutions. We take some local  $7 \times 7$  image patches from the images at different scales shown in Figure (10.3). These local image patches exhibit very different characteristics. Patches from near distance images are highly structured, corresponding to simple regular structures such as edges and corners, etc. As the distance increases, the patches become more irregular and random. So the local analysis in a computer vision system should be prepared to deal with such local image patches with different regularities and randomness.

### Change of compression rate

We perform some empirical studies on the change of statistical properties of the image data over scale. What we care most is the complexity or randomness of the image, and we measure the complexity rate or randomness empirically by JPEG 2000 compression rate. Generally speaking, for a simple and regular image, there are a lot of redundancies in the image intensities. So only a small number of bits are needed to store the image without any loss of information up to the discretization precision. For a complex and

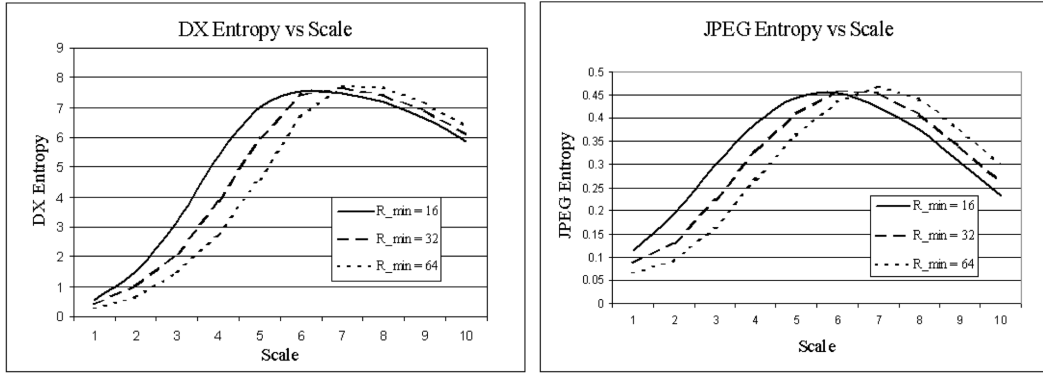


Figure 10.4: Reprinted with permission from [235]. The change of statistical properties over scale. (a) JPEG compression rate. (b) Entropy of marginal histogram of  $\nabla_x \mathbf{I}$ .

random image, there isn't much regularity or redundancy in the data. Therefore, a large number of bits are required to store the image. The reason we use JPEG 2000 compression rate to measure the complexity rate is two folded. First, JPEG 2000 is the state of the art image compression standard, and currently gives the best approximation to image complexity. Second, given the popularity of JPEG 2000, our results should also be interesting to image compression community.

When the image is compressed by JPEG 2000, the size of the compressed image file is recorded in terms of the number of bits. This number is then divided by the number of pixels to give the compression rate in terms of bits per pixel. Figure (10.4.a) plots this measurement in the order of viewing distance for images in Figure (10.2). At close distance, the randomness is small, meaning that the image is quite regular. Then the randomness starts to increase over distance, because more and more leaves are covered by the scope of the camera. At far distance, however, the randomness begins to decrease, because the local averaging operation reduces the marginal variance, and eventually smoothes the image into a constant image because of the law of large number. In this plot, there are three curves. They correspond to three different  $r_{\min}$  in our simulation study, while  $r_{\max}$  is always fixed at the same value. For smaller  $r_{\min}$ , the corresponding curve shifts to the left, because the average size of the leaves is smaller.

We also use a simple measure of smoothness as an indicator of randomness or complexity rate. We compute pairwise differences between intensities of adjacent pixels  $\nabla_x \mathbf{I}(i, j) = \mathbf{I}(i, j) - \mathbf{I}(i - 1, j)$  and  $\nabla_y \mathbf{I}(i, j) = \mathbf{I}(i, j) - \mathbf{I}(i, j - 1)$ .  $\nabla \mathbf{I}(i, j) = (\nabla_x \mathbf{I}(i, j), \nabla_y \mathbf{I}(i, j))$  is the gradient of  $\mathbf{I}$  at  $(i, j)$ . The gradient is a very useful local feature that can be used for edge detection [235]. It is also extensively used in image processing. We make a marginal histogram of  $\{\nabla_x \mathbf{I}(i, j), (i, j) \in \Lambda\}$  and compute the entropy of the histogram. Figure (10.4.b) plots this entropy over the order of distance for images in Figure (10.2). The plot behaves similarly as the plot of the JPEG 2000 compression rate.

## Variance normalization

The local averaging operation in Equation (10.1) reduces the marginal variance of the image intensities. A more appropriate measurement of randomness should be the compression rate of variance-normalized image, which is invariant of linear transformations of image intensities. Specifically, for an image  $\mathbf{I}$ , let  $\sigma^2$  be the marginal variance of  $\mathbf{I}$ . Let  $\mathbf{I}'(i, j) = \mathbf{I}(i, j)/\sigma$ . Then  $\mathbf{I}'$  is the variance-normalized version of  $\mathbf{I}$ , and the marginal variance of  $\mathbf{I}'$  is 1. We compute the JPEG compression rates of variance-normalized versions of the images in Figure (10.2). Figure (10.5.a) displays the variance-normalized JPEG compression rate over the order of distance for the three runs of the simulation study. The compression rate increases

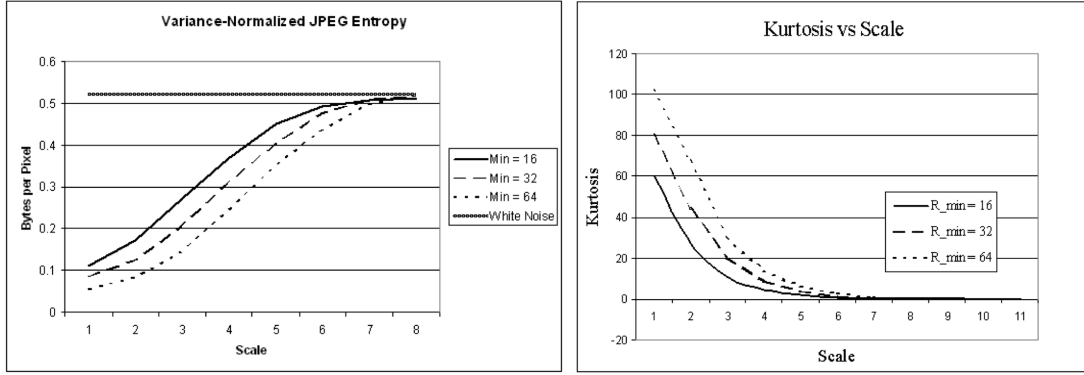


Figure 10.5: Reprinted with permission from [235]. (a) The change of JPEG compression rate of the variance-normalized versions of the images in Figure (10.2). (b) The change of kurtosis.

monotonically towards an upper bound represented by the horizontal line. This suggests that the scaling process increases the randomness and transforms a regular image to a random image. The upper bound is the JPEG compression rate of the Gaussian white noise process with variance 1.

The convergence of the compression rate of the variance-normalized image to that of the Gaussian white noise image is due to the effect of the central limit theorem. As another illustration, we compute the kurtosis of the marginal distribution of  $\{\nabla_x \mathbf{I}(x), x \in D\}$ . The kurtosis decreases monotonically towards 0, meaning that the image feature becomes closer to Gaussian distribution.

### Basic information theoretical concepts

Let  $\mathbf{I}(x, y)$  be an image with  $(x, y) \in D$ , where  $D$  is the discrete lattice of pixels (in what follows, we use  $(x, y)$  instead of  $(i, j)$  to denote discrete pixels). Let  $p(\mathbf{I})$  be the distribution of  $\mathbf{I}$ . We are interested in the following statistical properties [28].

1) *Entropy and entropy rate*: The entropy of  $p$  is defined as

$$\mathcal{H}(p) = \mathcal{E}_p[-\log p(\mathbf{I})] = - \int p(\mathbf{I}) \log p(\mathbf{I}) d\mathbf{I},$$

and the entropy rate of  $p$  is defined as  $\bar{\mathcal{H}}(p) = \mathcal{H}(p)/|D|$ , where  $|D|$  is the number of pixels in lattice  $D$ .

2) *Relative entropy and relative entropy rate*: For two distributions  $p$  and  $q$ , the relative entropy or the Kullback-Leibler divergence between  $p$  and  $q$  is defined as

$$\text{KL}(p||q) = \mathcal{E}_p \left[ \log \frac{p(\mathbf{I})}{q(\mathbf{I})} \right] = -\mathcal{H}(p) - \mathcal{E}_p[\log q(\mathbf{I})] \geq 0.$$

The relative entropy rate is  $\hat{k}(p||q) = \text{KL}(p||q)/|D|$ .

3) *Relative entropy with respect to Gaussian white noise*: For an image distribution  $p$ , let

$$\frac{1}{|D|} \sum_{(x,y) \in D} \mathcal{E}[\mathbf{I}(x, y)^2] = \sigma^2$$

be the marginal variance. Let  $q$  be the Gaussian white noise distribution with mean 0 and variance  $\sigma^2$ , i.e.,  $\mathbf{I}(x, y) \sim \mathcal{N}(0, \sigma^2)$  independently. Then

$$\text{KL}(p||q) = -\mathcal{H}(p) - \mathcal{E}_p[\log q(\mathbf{I})] = \mathcal{H}(q) - \mathcal{H}(p) \geq 0. \quad (10.4)$$



The second equation in (10.4) follows from  $\mathcal{E}_p[\log q(\mathbf{I})] = \mathcal{E}_q[\log q(\mathbf{I})]$  because  $\log q(\mathbf{I})$  is linear in  $\sum_{x,y} \mathbf{I}(x,y)^2$ , which has the same expectations under both  $p$  and  $q$ . Because  $\mathcal{H}(q) \geq \mathcal{H}(p)$  according to (10.4), the Gaussian white noise distribution has the maximum entropy among all the image distributions with the same marginal variance.

4) *Entropy rate of variance-normalized image:* Continue from (10.4) and calculate the entropy rate of Gaussian white noise explicitly, we obtain the relative entropy rate

$$\hat{k}(p||q) = \log \sqrt{2\pi e} - [\bar{\mathcal{H}}(p(\mathbf{I})) - \log \sigma] = \log \sqrt{2\pi e} - \bar{\mathcal{H}}(p(\mathbf{I}')),$$

where  $\mathbf{I}' = \mathbf{I}/\sigma$  is the variance-normalized version of image  $\mathbf{I}$ , and  $p(\mathbf{I}')$  denotes the distribution of  $\mathbf{I}'$ . So the entropy rate of the variance-normalized image  $\bar{\mathcal{H}}(p(\mathbf{I}'))$  determines the relative entropy rate  $\hat{k}(p||q)$  of  $p(\mathbf{I})$  with respect to the Gaussian white noise  $q(\mathbf{I})$ . In other words,  $\bar{\mathcal{H}}(p(\mathbf{I}'))$  measures the departure of  $p$  from the Gaussian white noise hypothesis.

## Change of entropy rate

For simplicity, let's study what happens if we double the viewing distance or zoom out the image by a factor of 2. Suppose the current image is  $\mathbf{I}(x,y)$ ,  $(x,y) \in \Lambda$ . If we double the viewing distance, the window covered by a pixel will double its size. So the original  $\mathbf{I}$  will be reduced to a smaller image  $\mathbf{I}_-$  defined on a reduced lattice  $\Lambda_-$ , and each pixel of  $\mathbf{I}_-$  will be the block average of four pixels of  $\mathbf{I}$ . More specifically, the process can be accounted for by two steps, similar to Equations (10.2) and (10.3).

(1) *Local smoothing:* Let the smoothed image be  $\mathbf{J}$ , then  $\mathbf{J}(x,y) = \sum_{u,v} \mathbf{I}(x+u, y+v)/4$ , where  $(u,v) \in \{(0,0), (0,1), (1,0), (1,1)\}$ . We can write  $\mathbf{J} = \mathbf{I} * g$  where  $g$  is the uniform distribution over  $\{(0,0), (0,-1), (-1,0), (-1,-1)\}$ . In general,  $g$  can be any kernel with appropriate bandwidth, such as a Gaussian distribution function.

(2) *Subsampling:*  $\mathbf{I}_-^{(u,v)}(x,y) = \mathbf{J}(2x+u, 2y+v)$ , where, again,  $(u,v) \in \{(0,0), (0,1), (1,0), (1,1)\}$ . Any of the four  $\mathbf{I}_-^{(u,v)}$  can be regarded as a subsampled version of  $\mathbf{J}$ .

**Theorem 2.** *Smoothing effect: Let  $\Lambda$  be an  $M \times N$  lattice, and  $\mathbf{I}$  is defined on  $D$ . Let  $\mathbf{J} = \mathbf{I} * g$ , where  $g$  is a local averaging kernel or a probability distribution. As  $\min(M, N) \rightarrow \infty$ ,*

$$\bar{\mathcal{H}}(p(\mathbf{J})) - \bar{\mathcal{H}}(p(\mathbf{I})) \rightarrow \frac{1}{4\pi^2} \int_0^{2\pi} \int_0^{2\pi} \log |\hat{g}(\omega)| d\omega \leq 0, \quad (10.5)$$

where  $\omega = (\omega_x, \omega_y)$  is the spatial frequency, and  $\hat{g}(\omega) = \sum_{x,y} g(x,y) \exp\{-i(\omega_x x + \omega_y y)\}$  is the Fourier transform of the kernel  $g$ , where the sum is over the support of  $g$ .

**Proof:** Let  $\mathbf{I}$  be the image defined on the integer lattice  $[0, M-1] \times [0, N-1]$ . The discrete Fourier transform of  $\mathbf{I}$  is

$$\hat{\mathbf{I}}(\omega) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} \mathbf{I}(x,y) \exp\{-i(\omega_x x + \omega_y y)\},$$

where  $\omega_x \in \{2\pi m/M, m = 0, \dots, M-1\}$  and  $\omega_y \in \{2\pi n/N, n = 0, \dots, N-1\}$ . The Fourier transforms of  $\mathbf{J}$  and  $g$  can be similarly defined. Because  $\hat{\mathbf{I}}$  and  $\hat{\mathbf{J}}$  are obtained from  $\mathbf{I}$  and  $\mathbf{J}$  respectively by the same linear transformation,  $\mathcal{H}(p(\hat{\mathbf{J}})) - \mathcal{H}(p(\hat{\mathbf{I}})) = \mathcal{H}(p(\mathbf{J})) - \mathcal{H}(p(\mathbf{I}))$ .

For convolution with periodic boundary condition,  $\hat{\mathbf{J}}(\omega) = \hat{\mathbf{I}}(\omega)\hat{g}(\omega)$ . So

$$\begin{aligned}\bar{\mathcal{H}}(p(\mathbf{J})) - \bar{\mathcal{H}}(p(\mathbf{I})) &= \frac{1}{|D|} \left[ \mathcal{H}(p(\hat{\mathbf{J}})) - \mathcal{H}(p(\hat{\mathbf{I}})) \right] \\ &= \frac{1}{MN} \sum_{\omega} \log |\hat{g}(\omega)| = \frac{1}{4\pi^2} \sum_{\omega} \log |\hat{g}(\omega)| \Delta\omega \\ &\rightarrow \frac{1}{4\pi^2} \int_0^{2\pi} \int_0^{2\pi} \log |\hat{g}(\omega)| d\omega,\end{aligned}$$

as  $\min(M, N) \rightarrow \infty$ , where  $\Delta\omega = (2\pi/M) \times (2\pi/N)$ .

A smoothing kernel  $g$  is a probability distribution function,  $\hat{g}$  is the characteristic function of  $g$ , and

$$\begin{aligned}\hat{g}(\omega) &= \sum_{x,y} g(x, y) \exp\{-i(\omega_x x + \omega_y y)\} \\ &= \mathcal{E}_g [\exp\{-i(\omega_x X + \omega_y Y)\}],\end{aligned}$$

where  $(X, Y) \sim g(x, y)$ . Then,

$$\begin{aligned}|\hat{g}(\omega)|^2 &= |\mathcal{E}_g [\exp\{-i(\omega_x X + \omega_y Y)\}]|^2 \\ &\leq \mathcal{E}_g [|\exp\{-i(\omega_x X + \omega_y Y)\}|^2] = 1.\end{aligned}$$

Thus,  $\int \log |\hat{g}(\omega)| d\omega \leq 0$ . QED

The above theorem tells us that there is always loss of information under the smoothing operation. This is consistent with the intuition in scale space theory, where the increase in scale results in the loss of fine details in the image. The change of entropy rate under linear filtering was first derived in the classical paper of Shannon (1948) [198].

Next, let's study the effect of subsampling. There are four subsampled versions  $\mathbf{I}_-^{(u,v)}(x, y) = \mathbf{J}(2x + u, 2y + v)$ , where  $(u, v) \in \{(0, 0), (0, 1), (1, 0), (1, 1)\}$ . Each  $\mathbf{I}_-^{(u,v)}$  is defined on a subsampled lattice  $\Lambda_-$ , with  $|\Lambda_-| = |\Lambda|/4$ .

**Theorem 3.** *Subsampling effect: The average entropy rate of  $\mathbf{I}_-^{(u,v)}$  is no less than the entropy rate of  $\mathbf{J}$ ,*

$$\frac{1}{4} \sum_{u,v} \bar{\mathcal{H}}(p(\mathbf{I}_-^{(u,v)})) - \bar{\mathcal{H}}(p(\mathbf{J})) = \bar{\mathcal{M}}(\mathbf{I}_-^{(u,v)}, \forall(u, v)) \geq 0, \quad (10.6)$$

where  $\mathcal{M}(\mathbf{I}_-^{(u,v)}, \forall(u, v)) = \text{KL}(p(\mathbf{J}) || \prod_{u,v} p(\mathbf{I}_-^{(u,v)}))$  is defined as the mutual information among the four subsampled versions, and  $\bar{\mathcal{M}} = \mathcal{M}/|D|$ .

**Proof:**

$$\begin{aligned}\sum_{u,v} \mathcal{H}(p(\mathbf{I}_-^{(u,v)})) - \mathcal{H}(p(\mathbf{J})) &= \mathcal{E} \left[ \log \frac{p(\mathbf{J})}{\prod_{u,v} p(\mathbf{I}_-^{(u,v)})} \right] \\ &= \text{KL}(p(\mathbf{J}) || \prod_{u,v} p(\mathbf{I}_-^{(u,v)})) \\ &= \mathcal{M}(\mathbf{I}_-^{(u,v)}, \forall(u, v)) \geq 0,\end{aligned}$$

where the expectation is with respect to the distribution of  $\mathbf{J}$ , which is also the joint distribution of  $\mathbf{I}_-^{(u,v)}$ . QED

The scaling of the entropy rate is a combination of Equations (10.5) and (10.6):

$$\left\{ \frac{1}{4} \sum_{u,v} \bar{\mathcal{H}}(p(\mathbf{I}_{-}^{(u,v)})) - \bar{\mathcal{H}}(p(\mathbf{I})) \right\} - \left\{ \bar{\mathcal{M}}(\mathbf{I}_{-}^{(u,v)}) + \frac{1}{4\pi^2} \int \log |\hat{g}(\omega)| d\omega \right\} \rightarrow 0. \quad (10.7)$$

For regular image patterns, the mutual information per pixel can be much greater than  $-\int \log |\hat{g}(\omega)| d\omega / 4\pi^2$ , so the entropy rate increases with distance, or in other words, the image becomes more random. For very random patterns, the reverse is true. When the mutual information rate equals to  $-\int \log |\hat{g}(\omega)| d\omega / 4\pi^2$ , we have scale invariance. More careful analysis is needed to determine when this is true.

Next we study the change of entropy rate of variance-normalized image  $\bar{\mathcal{H}}(p(\mathbf{I}'))$ . For simplicity, let's assume that  $p(\mathbf{I})$  comes from a stationary process, and  $\mathbf{I}_{-}$  can be any sub-sampled version of  $\mathbf{J} = \mathbf{I} * g$ , which is also stationary. Let  $\sigma^2 = \text{Var}[\mathbf{I}(x, y)]$  and  $\sigma_{-}^2 = \text{Var}[\mathbf{I}_{-}(x, y)]$  be the marginal variances of  $\mathbf{I}$  and  $\mathbf{I}_{-}$  respectively. Let  $\mathbf{I}' = \mathbf{I}/\sigma$  and  $\mathbf{I}'_{-} = \mathbf{I}_{-}/\sigma_{-}$  be the variance-normalized versions of  $\mathbf{I}$  and  $\mathbf{I}_{-}$  respectively. It is easy to show that

$$\rho^2 = \frac{\sigma_{-}^2}{\sigma^2} = \frac{1}{4} \sum_{u,v} \text{corr}(\mathbf{I}(x, y), \mathbf{I}(x+u, y+v)) \leq 1, \quad (u, v) \in \{(0, 0), (0, 1), (1, 0), (1, 1)\},$$

so the smoothing operation reduces the marginal variance. Therefore, we can modify (10.7) into

$$\bar{\mathcal{H}}(p(\mathbf{I}'_{-})) - \bar{\mathcal{H}}(p(\mathbf{I}')) \approx \bar{\mathcal{M}}(\mathbf{I}_{-}^{(u,v)}) - \log \rho + \frac{1}{4\pi^2} \int \log |\hat{g}(\omega)| d\omega, \quad (10.8)$$

where the difference between left-hand side and right-hand side converges to 0 as  $|D| \rightarrow \infty$ . In (10.8), the term  $-\log \rho$  is positive, and it compensates for the loss of entropy rate caused by smoothing, i.e.,  $\int \log |\hat{g}(\omega)| d\omega / 4\pi^2$ , which is negative. As a matter of fact, the first two terms, i.e., the mutual information term and the  $-\log \rho$  term on the right hand side of (10.8) balance each other, in the sense that if one is small, then the other tends to be large. However, we have not been able to identify conditions under which the right-hand side of (10.8) is always positive, which would have established the monotone increase of the entropy rate of variance-normalized image or monotone decrease of the departure from Gaussian white noise.

## 10.2 Perceptual Entropy

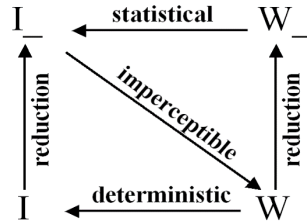


Figure 10.6: Reprinted with permission from [235]. Transition from deterministic representation to statistical description.

The above analysis on entropy rate is only about the observed image  $\mathbf{I}$  alone. The goal of computer vision is to interpret the observed image in order to recognize the objects in the outside world. In this subsection, we shall go beyond the statistical properties of the observed image itself, and study the interaction between the observed image and the outside scene that produces the image.

Again, we would like to use the dead leaves model to convey the basic idea. Suppose our attention is restricted to a finite scope  $\Omega \subset \mathbf{R}^2$ , and let  $W = ((x_i, y_i, r_i, a_i), i = 1, \dots, N)$  be the leaves in  $\Omega$  that are not completely occluded by other leaves. Then we have  $W \sim p(W)$  and  $\mathbf{I} = \gamma(W)$ , where  $p(W)$  comes from the Poisson process that generates the dead leaves, and  $\gamma$  represents the transformation defined by Equation (10.1) for a scale parameter  $s$ .

For convenience, we assume that both  $W$  and  $\mathbf{I}$  are properly discretized. For any joint distribution  $p(W, \mathbf{I})$ , the conditional entropy  $\mathcal{H}(p(W | \mathbf{I}))$  is defined as

$$\mathcal{H}(p(W | \mathbf{I})) = - \sum_{W, \mathbf{I}} p(W, \mathbf{I}) \log p(W | \mathbf{I}). \quad (10.9)$$

$\mathcal{H}(p(W | \mathbf{I}))$  measures the inferential uncertainty or imperceptibility of  $W$  from the image  $\mathbf{I}$ .

**Proposition 1.** *If  $W \sim p(W)$  and  $\mathbf{I} = \gamma(W)$ , then  $\mathcal{H}(p(W | \mathbf{I})) = \mathcal{H}(p(W)) - \mathcal{H}(p(\mathbf{I}))$ . That is, imperceptibility = scene entropy - image entropy.*

This proposition is easy to prove. The marginal distribution of  $\mathbf{I}$  is  $p(\mathbf{I}) = \sum_{W: \gamma(W)=\mathbf{I}} p(W)$ . The posterior distribution of  $W$  given  $\mathbf{I}$  is  $p(W | \mathbf{I}) = p(W, \mathbf{I}) / p(\mathbf{I}) = p(W) / p(\mathbf{I})$ . Here  $p(W, \mathbf{I}) = p(W)$  because  $\mathbf{I}$  is determined by  $W$ . Following the definition in (10.9),  $\mathcal{H}(p(W | \mathbf{I})) = - \sum_W p(W) (\log p(W) - \log p(\mathbf{I})) = \mathcal{H}(p(W)) - \mathcal{H}(p(\mathbf{I}))$ . Here  $\mathcal{E}_W[\log p(\mathbf{I})] = \mathcal{E}_{\mathbf{I}}[\log p(\mathbf{I})]$  since  $\mathbf{I}$  is determined by  $W$ .

If we increase the viewing distance or equivalently zooming out the camera while fixing the scope  $\Omega \subset \mathbf{R}^2$ , i.e., fixing  $W$ , then we obtain a zoomed-out version  $\mathbf{I}_- = R(\mathbf{I})$ , where  $R$  represents the zooming-out operation of smoothing and subsampling, and is a many-to-one transformation. During the process of zooming out, the total entropy of the image will decrease, i.e.,  $\mathcal{H}(p(\mathbf{I}_-)) \leq \mathcal{H}(p(\mathbf{I}))$ , even though the entropy per pixel can increase as we have shown in the previous subsection. Therefore, we have the following result.

**Proposition 2.** *If  $W \sim p(W)$ ,  $\mathbf{I} = \gamma(W)$ , and  $\mathbf{I}_- = R(\mathbf{I})$ , where  $R$  is a many-to-one mapping, then  $\mathcal{H}(p(W | \mathbf{I}_-)) \geq \mathcal{H}(p(W | \mathbf{I}))$ , i.e., the imperceptibility increases as the image is reduced.*

What does this result tell us in terms of interpreting image  $\mathbf{I}$  or  $\mathbf{I}_-$ ? Although the model  $W \sim p(W)$  and  $\mathbf{I} = \gamma(W)$  is the right physical model for all the scale  $s$ , this model is meaningful in interpreting  $\mathbf{I}$  only within a limited range, say  $s \leq s_{\text{bound}}$ , so that the imperceptibility  $\mathcal{H}(p(W | \mathbf{I}))$  is below a small threshold. In this regime, the representation  $\mathbf{I} = \gamma(W)$  is good for both recognition and coding. For recognition,  $\mathcal{H}(p(W | \mathbf{I}))$  is small, so  $W$  can be accurately determined from  $\mathbf{I}$ . For coding, we can first code  $W$  according to  $p(W)$ , with a coding cost  $\mathcal{H}(p(W))$ . Then we code  $\mathbf{I}$  using  $\mathbf{I} = \gamma(W)$  without any coding cost. The total coding cost would be just  $\mathcal{H}(p(W))$ . If the imperceptibility  $\mathcal{H}(p(W | \mathbf{I}))$  is small,  $\mathcal{H}(p(W)) \approx \mathcal{H}(p(\mathbf{I}))$ , so coding  $W$  will not incur coding overhead.

But if  $s$  is very large, the imperceptibility  $\mathcal{H}(p(W | \mathbf{I}))$  can be large according to Proposition 2. In this case, the representation  $\mathbf{I} = \gamma(W)$  is not good for either recognition or coding. For recognition,  $W$  cannot be estimated with much certainty. For coding, if we still code  $W$  first, and code  $\mathbf{I}$  by  $\mathbf{I} = \gamma(W)$ , this will not be an efficient coding, since  $\mathcal{H}(p(W))$  can be much larger than  $\mathcal{H}(p(\mathbf{I}))$ , and the difference is imperceptibility  $\mathcal{H}(p(W | \mathbf{I}))$ .

Then what should we do? The regime of  $s > s_{\text{bound}}$  is quite puzzling for vision modeling. Our knowledge about geometry, optics, and mechanics enables us to model every phenomenon in our physical

environment. Such models may be sufficient for computer graphics as far as generating physically realistic images is concerned. For instance, a garden scene can be constructed by simulating billions of leaves and grass strands, and the image can be produced by projecting these billions of objects onto the image with perspective geometry. A river scene, a fire scene or a smoke scene can be obtained using computational fluid dynamics. A piece of cloth can be generated using a dense set of particles that follow the law of mechanics. Realistic lighting can be simulated by ray tracing and optics. But such models are hardly meaningful for vision, because the imperceptibilities of the underlying elements or variables are intolerable. When we look at a garden scene, we never really perceive every leaf or every strand of grass. When we look at a river scene, we do not perceive the constituent elements used in fluid dynamics. When we look at a scene with sophisticated lighting and reflection, we do not trace back the light rays. In those situations where physical variables are not perceptible due to scaling or other aspects of image formation process, it is quite a challenge to come up with good models for the observed images. Such models do not have to be physically realistic, but they should generate visually realistic images, so that such models can be employed to interpret the observed image at a level of sophistication that is comparable to human vision.

The following are some of our simple theoretical considerations of this problem from the perspectives of recognition and coding. We shall become more concrete on the modeling issue in subsequent sections.

Suppose the image  $\mathbf{I}$  is reduced to an image  $\mathbf{I}_- = R(\mathbf{I})$ , so that  $W$  cannot be reliably inferred. Then, instead of pursuing a detailed description  $W$  from  $\mathbf{I}_-$ , we may choose to estimate some aspects of  $W$  from  $\mathbf{I}_-$ . For instance, in the simulated ivy wall example, we may estimate properties of the overall distribution of colors of leaves, as well as the overall distribution of their sizes, etc. Let's call it  $W_- = \rho(W)$ , with  $\rho$  being a many to one reduction function. It is possible that we can estimate  $W_-$  from  $\mathbf{I}_-$  because of the following result.

**Proposition 3.** *Let  $W \sim p(W)$ ,  $\mathbf{I} = \gamma(W)$ , and  $W_- = \rho(W)$ ,  $\mathbf{I}_- = R(\mathbf{I})$ , where both  $\rho$  and  $R$  are many to one mappings, we have*

$$(1) \quad \mathcal{H}(p(W_-|\mathbf{I}_-)) \leq \mathcal{H}(p(W|\mathbf{I}_-)).$$

$$(2) \quad p(\mathbf{I}_-|W_-) = \frac{\sum_{W:\rho(W)=W_-; R(\gamma(W))=\mathbf{I}_-} p(W)}{\sum_{W:\rho(W)=W_-} p(W)}.$$

Result (1) tells us that even if  $W$  is imperceptible from  $\mathbf{I}_-$ ,  $W_-$  may still be perceptible. Result (2) tells us that although  $W$  defines  $\mathbf{I}$  deterministically via  $\mathbf{I} = \gamma(W)$ ,  $W_-$  may only define  $\mathbf{I}_-$  statistically via a probability distribution  $p(\mathbf{I}_-|W_-)$ . While  $W$  represents deterministic structures,  $W_-$  may only represent some texture properties. Thus, we have a transition from a deterministic representation of the image intensities  $\mathbf{I} = \gamma(W)$  to a statistical characterization  $\mathbf{I}_- \sim p(\mathbf{I}_-|W_-)$ . See Figure (10.6) for an illustration.

For an image  $\mathbf{I}$ , we may extract  $F(\mathbf{I})$ , which can be a dimension reduction or a statistical summary, so that  $F(\mathbf{I})$  contains as much information about  $\mathbf{I}$  as possible as far as  $W$  or  $W_-$  is concerned. In the following proposition, we shall not distinguish between  $(W, \mathbf{I})$  and  $(W_-, \mathbf{I}_-)$  for notational uniformity.

**Proposition 4.** *Let  $F = F(\mathbf{I})$ ,*

*(1) If  $W \sim p(W)$ ,  $\mathbf{I} = \gamma(W)$ , then  $\text{KL}(p(W|\mathbf{I})||p(W|F)) = \mathcal{H}(p(\mathbf{I}|F))$ .*

*(2) If  $W \sim p(W)$  and  $[\mathbf{I}|W] \sim p(\mathbf{I}|W)$ , then  $\text{KL}(p(W|\mathbf{I})||p(W|F)) = \mathcal{M}(W, \mathbf{I}|F)$ , where  $\mathcal{M}(W, \mathbf{I}|F) = \mathcal{E}_{W, \mathbf{I}} \{\log[p(W, \mathbf{I}|F)/(p(W|F)p(\mathbf{I}|F))]\}$  is the mutual information between  $W$  and  $\mathbf{I}$  given  $F$ .*

Result (1) tells us that for  $F(\mathbf{I})$  to contain as much information about  $W$  as possible, we want to make  $\mathcal{H}(p(\mathbf{I}|F))$  to be as small as possible, so that  $F$  can be used to reconstruct  $\mathbf{I}$  accurately. Result (2) tells us that if we want to estimate  $W$ , we want  $F$  to be sufficient about  $\mathbf{I}$  as far as  $W$  is concerned.  $\mathcal{M}(W, \mathbf{I}|F)$  can be considered a measurement of sufficiency.

Now let's study this issue from the coding perspective. Suppose the image  $\mathbf{I}$  follows a true distribution  $f(\mathbf{I})$ , and we use a model  $w \sim p(w)$ , and  $[\mathbf{I} \mid w] \sim p(\mathbf{I} \mid w)$  to code  $\mathbf{I} \sim f(\mathbf{I})$ . Here the variable  $w$  is augmented solely for the purpose of coding. It might be some  $w = W_- = \rho(W)$ , or it may not have any correspondence to the reality  $W$ . In the coding scheme, for an image  $\mathbf{I}$ , we first estimate  $w$  by a sample from the posterior distribution  $p(w|\mathbf{I})$ , then we code  $w$  by  $p(w)$  with coding length  $-\log p(w)$ . After that, we code  $\mathbf{I}$  by  $p(\mathbf{I}|w)$  with coding length  $-\log p(\mathbf{I}|w)$ . So the average coding length is  $-\mathcal{E}_f [\mathcal{E}_{p(w|\mathbf{I})}(\log p(w) + \log p(\mathbf{I}|w))]$ .

**Proposition 5.** *The average coding length is  $\mathcal{E}_f[\mathcal{H}(p(w|\mathbf{I}))] + \text{KL}(f(\mathbf{I})||p(\mathbf{I})) + \mathcal{H}(f)$ , where  $p(\mathbf{I}) = \sum_w p(w)p(\mathbf{I} \mid w)$  is the marginal distribution of  $\mathbf{I}$  under the model. So, coding redundancy = imperceptibility + model bias.*

The above proposition provides a selection criterion for models with latent variables. The imperceptibility term comes up because we assume a coding scheme where  $w$  must be coded first, and then  $\mathbf{I}$  is coded based on  $w$ . Given the latent variable structure of the model, it is very natural to assume such a coding scheme.

### 10.3 A Continuous Spectrum



Figure 10.7: The image contains patterns of different complexities, from very simple patterns such as geometric patterns to very random patterns such as leaves at far distance.

Image patterns of different entropy regimes are not only connected by image scaling, they co-exist and blend seamlessly in a single image. For instance, imagine we are in a wood of maple trees and taking a picture. The patterns displayed in Fig.10.1 may appear together in the picture we take, because the maple leaves can appear at different distances from the camera when the picture is taken. In addition, even for the same objects in a fixed image, when we analyze this image in multiple resolutions, we may recognize patterns from different regimes. The close connection between different regimes calls for a common theoretical framework for modeling patterns in these regimes. In particular, it calls for the integration of the MRFs and sparse coding models that work well in high-entropy regime and low-entropy regime respectively.

### 10.4 Two Coding Schemes

In this section, we shall examine two concrete classes of image models and analyze their entropy behaviors.

The image coding theories can be traced back to Fourier and harmonic analysis, where  $\mathbf{I}$  is represented as a linear combination of a dictionary of image bases  $\mathbf{\Gamma} = \{\Gamma_i, i = 1, \dots, N\}$ :

$$\mathbf{I} = \sum_{i=1}^N c_i \Gamma_i + \epsilon, \quad \Gamma_i \in \mathbf{\Gamma}, \quad C = \{c_i\} \sim p(C), \quad (10.10)$$

where  $c_i$  are coefficients,  $\epsilon$  is the residual, and  $\epsilon \sim \text{iid } N(0, \sigma^2)$ . If  $\mathbf{\Gamma}$  consists of orthogonal sine waves, then the model reduces to Fourier analysis. Many other dictionaries have been proposed in the literature and they can be used in combination. The dictionaries are often over-complete, i.e., for an image  $\mathbf{I}$  defined on lattice  $D$ ,  $N \gg |D|$ . In the proposed work,  $\mathbf{\Gamma}$  is a collection of Gabor wavelets  $\{G_{x,s,\theta}\}$ , so  $i$  indexes  $(x, s, \theta)$ , i.e., (location, scale, orientation).

For over-complete  $\mathbf{\Gamma}$ , the coefficients  $C = (c_i, i = 1, \dots, N)$  cannot be uniquely determined. The guiding principle for inferring the coefficients is *sparsity*, which holds that for each image  $\mathbf{I}$ , only a small number of coefficients should be significantly different from 0. The sparsity can be modeled by probability distributions with probability mass concentrated at 0, but with long tails to account for occasionally large values, e.g.,  $p(C) = \prod_i p(c_i)$ ,  $p(c_i) \propto e^{-\alpha|c_i|}$ . The sparsity can also be measured by norms such as the weak  $\ell^p$ -norm.

The above characterizations of sparsity are continuous without explicit selection. For the purpose of image modeling, we need to select a small subset  $\{B_k, k = 1, \dots, K\} \subset \mathbf{\Gamma}$  to represent  $\mathbf{I}$ , so that the projection of  $\mathbf{I}$  onto the subspace spanned by  $\{B_k\}$  explains as much variation in  $\mathbf{I}$  as possible. Here we would like to clarify that each  $B_k$  is a  $\Gamma_i$  for some  $i \in \{1, \dots, N\}$ . If we use Gabor bases, then each  $B_k$  is a  $G_{x,s,\theta}$  for some  $(x, s, \theta)$ . Because of sparsity,  $K \ll |D| \ll N$ . This is essentially the variable selection problem in linear regression.

There are two problems with sparse coding though: 1) The textures are not well represented. 2) The pattern formed by the selected bases are not modeled. The model has no concepts such as lines.

The sparse coding model is limited to low-entropy regime. We can rewrite model (10.10) in matrix form  $C \sim p(C)$ ,  $\mathbf{I} = \mathbf{J} + \epsilon$ , and  $\mathbf{J} = \mathbf{\Gamma}C$ , where  $\mathbf{I}$  and  $\mathbf{J}$  become vectors,  $\mathbf{\Gamma}$  is the matrix composed of all the bases  $\{\Gamma_i\}$ , and  $C$  is the vector consisting of all the  $\{c_i\}$ . For a distribution  $p(x)$ , define its entropy by  $\mathcal{H}(p) = -\int p(x) \log p(x) dx$  [28]. If  $p(C)$  is very sparse, then  $\mathcal{H}(p(C))$  is small.

**Proposition 6.** *Let  $p(\mathbf{J})$  be the distribution of  $\mathbf{J} = \mathbf{\Gamma}C$ ,  $\mathcal{H}(p(\mathbf{J})) \leq \mathcal{H}(p(C)) + \log |\det(\mathbf{\Gamma}\mathbf{\Gamma}')|/2$ .*

In contrast to sparse coding models that represent the image explicitly, the MRF models summarize the image by some feature statistics. For instance, when one views the leaves at far distance, one gathers a texture impression without identifying the individual leaves. One has to drop many variables such as the positions and shapes of the individual leaves as they cannot be inferred unambiguously, and pool together some spatial statistics over the image. It has long been observed in psychophysics that human visual perception cannot distinguish two texture regions if they share certain spatial statistics.

Recall the dictionary of Gabor wavelets  $\{G_{x,s,\theta}\}$ . In the sparse coding model, a small number of  $G_{x,s,\theta}$  with specific locations  $x$  are selected, and they are denoted by  $\{B_k, k = 1, \dots, K\}$ . In contrast, in texture model, if a filter  $G_{s,\theta}$  is selected, it is applied to every pixel  $x \in D$  to get the responses  $[G_{s,\theta} * \mathbf{I}](x) = \langle \mathbf{I}, G_{x,s,\theta} \rangle$ , and these responses are to be pooled over all  $x \in D$  into a marginal histogram, while discarding the location information.

Specifically, let  $\{G_m, m = 1, \dots, M\}$  be a set of filters selected to model an image  $\mathbf{I}$ , where each  $G_m$  is a  $G_{s,\theta}$  for some  $(s, \theta)$ . By pooling the filter responses  $[G_m * \mathbf{I}](x) = \langle \mathbf{I}, G_{x,m} \rangle$  over  $x \in D$ , one obtains a histogram. Specifically, we divide the range of  $[G_m * \mathbf{I}](x)$  into  $T$  bins  $\Delta_1, \dots, \Delta_T$ , so that  $H_{m,t}(\mathbf{I}) = \sum_{x \in D} 1_{[G_m * \mathbf{I}](x) \in \Delta_t}$  counts the number of  $[G_m * \mathbf{I}](x)$  that fall into the  $t$ -th bin,  $t = 1, \dots, T$ . Let

$h_{m,t}(\mathbf{I}) = H_{m,t}(\mathbf{I})/|D|$  be the normalized marginal histogram. For simplicity, we write  $H_m = (H_{m,t}, t = 1, \dots, T)$ , and  $h_m = (h_{m,t}, t = 1, \dots, T)$ . Wu, Zhu, and Liu (2000) defined the following image ensemble:

$$\Omega = \{\mathbf{I} : h_m(\mathbf{I}) = h_m(\mathbf{I}_{\text{obs}}), m = 1, \dots, M\}. \quad (10.11)$$

If the set of marginal histograms  $\{h_m\}$  captures the texture information in  $\mathbf{I}_{\text{obs}}$ , then all the images in the ensemble  $\Omega$  should share the same texture pattern. So we can model the image  $\mathbf{I}_{\text{obs}}$  as a random sample from the uniform distribution over  $\Omega$ . In order to select a set of  $\{G_m, m = 1, \dots, M\}$  to model a texture image  $\mathbf{I}_{\text{obs}}$ , we want the resulting  $\Omega$  defined in (10.11) to have minimum volume  $|\Omega|$  or entropy  $\log |\Omega|$ , or to have maximum log-likelihood  $-\log |\Omega|$ .  $\log |\Omega|$  measures how well  $\{G_m, m = 1, \dots, M\}$  and their histograms explain  $\mathbf{I}_{\text{obs}}$ , and can be used as a criterion for selecting the most informative set of filters.

As observed by Wu, Zhu, and Liu (2000),  $\text{Unif}(\Omega)$  can be made equivalent to MRF model, thanks to two most profound results in statistical physics and information theory respectively.

1) According to *equivalence of ensembles* in statistical physics, for any fixed part of the image lattice  $D_0 \subset D$ , under  $\text{Unif}(\Omega)$ , as  $|D| \rightarrow \infty$ , the distribution of the image intensities on the fixed  $D_0$ ,  $\mathbf{I}_{D_0}$ , converges to the so-called FRAME (filters, random field, and maximum entropy) model developed by Zhu, Wu, and Mumford

$$p(\mathbf{I}_{D_0}; \Lambda) = \frac{1}{Z(\Lambda)} \exp\left\{\sum_1^M \langle \lambda_m, H_m(\mathbf{I}_{D_0}) \rangle\right\}, \quad (10.12)$$

where  $\Lambda = (\lambda_m, m = 1, \dots, M)$ ,  $\lambda_m = (\lambda_{m,t}, t = 1, \dots, T)$ , and  $Z(\Lambda)$  is the normalizing constant.

2) According to *asymptotic equipartition property* in information theory, as  $|D| \rightarrow \infty$ , the FRAME model  $p(\mathbf{I}_D; \Lambda)$  of the form defined by (10.12) (with  $D_0$  replaced by  $D$ ) concentrates its probability mass uniformly on the image ensemble  $\Omega$  of the form (10.11).

In the FRAME model  $p(\mathbf{I}; \Lambda)$ ,  $\Lambda$  can be estimated by maximizing the log-likelihood  $\log p(\mathbf{I}_{\text{obs}}; \Lambda)$ , which amounts to solving the estimating equation  $\mathcal{E}_\Lambda[H_m(\mathbf{I})] = H_m(\mathbf{I}_{\text{obs}})$  for  $m = 1, \dots, M$ . The entropy  $\log |\Omega|$  can be approximated by  $-\log p(\mathbf{I}_{\text{obs}}; \Lambda_{\text{MLE}})$ . The FRAME model extends the traditional clique-based MRF models by using filters and statistics that are relevant to vision. The feature statistics in FRAME models are not limited to marginal histograms of filter responses. It can be any spatial statistics pooled over  $x \in D$ .

The FRAME model only specifies the marginal distributions of filter responses, it cannot represent large regular structures such as lines. The FRAME model works well for the high-entropy regime. For two distributions, define the Kullback-Leibler divergence by  $\mathcal{H}(q|p) = \int \log(q(x)/p(x))q(x)dx$ .

**Proposition 7.** *Let  $p(\mathbf{I})$  be the FRAME model. For any distribution  $q(\mathbf{I})$  such that  $\mathcal{E}_q[h_m(\mathbf{I})] = \mathcal{E}_p[h_m(\mathbf{I})]$ ,  $\forall m$ , we have  $\mathcal{H}(q|p) = \mathcal{H}(p) - \mathcal{H}(q) \geq 0$ . So the FRAME model always approaches the entropy of the true distribution from above.*

## 10.5 Perceptual Scale Space

When an image is viewed at varying resolutions, it is known to create discrete perceptual jumps or transitions amid the continuous intensity changes. Wang and Zhu (2002) studied a *perceptual scale-space* theory which differs from the traditional *image scale-space* theory in two aspects. (i) In representation, the perceptual scale-space adopts a full generative model. From a Gaussian pyramid it computes a *sketch pyramid* where each layer is a primal sketch representation – an attribute graph whose elements are image primitives for the image structures. Each primal sketch graph generates the image in the Gaussian pyramid, and the



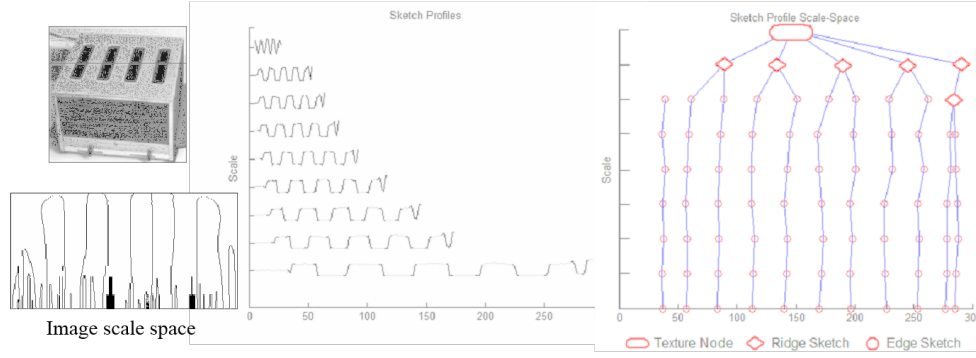


Figure 10.8: Reprinted with permission from [226]. Scale-space of a 1D signal. (a) A toaster image from which a line is taken as the 1D signal. (b) Trajectories of zero-crossings of the 2nd derivative of the 1D signal. The finest scale is at the bottom. (c) The 1D signal at different scales. The black segments on the curves correspond to primal sketch primitives (step edge or bar). (d) A symbolic representation of the sketch in scale-space with three types of transitions.

changes between the primal sketch graphs in adjacent layers are represented by a set of basic and composite *graph operators* to account for the perceptual transitions. (ii) In computation, the sketch pyramid and graph operators are inferred, as hidden variables, from the images through Bayesian inference by stochastic algorithm, in contrast to the deterministic transforms or feature extraction, such as computing zero-crossings, extremal points, and inflection points in the image scale-space. Studying the perceptual transitions under the Bayesian framework makes it convenient to use the statistical modeling and learning tools for (a) modeling the Gestalt properties of the sketch graph, such as continuity and parallelism etc; (b) learning the most frequent graph operators, i.e. perceptual transitions, in image scaling; and (c) learning the prior probabilities of the graph operators conditioning on their local neighboring sketch graph structures.

In experiments, they learn the parameters and decision thresholds through human experiments, and they show that the sketch pyramid is a more parsimonious representation than a multi-resolution Gaussian/Wavelet pyramid. They also demonstrate an application on adaptive image display – showing a large image in a small screen (say PDA) through a selective tour of its image pyramid. In this application, the sketch pyramid provides a means for calculating information gain in zooming-in different areas of an image by counting a number of operators expanding the primal sketches, such that the maximum information is displayed in a given number of frames.

## 10.6 Perceptibility, Metastability, and the Energy Landscape

The distribution of images in the image space  $\Omega_I \subset \mathbb{R}^N$  defined by an energy function  $U : \Omega_I \rightarrow \mathbb{R}$  can be understood as manifold, or landscape, of dimension  $\mathbb{R}^N$  in the high-dimensional space  $\mathbb{R}^{N+1}$ . The energy function is given by  $U = -\log f$ , where  $f$  is a density over  $\Omega_I$ . High-probability states of  $f$  are low-energy (stable) states of  $U$ , and low-probability states of  $f$  are high-energy (unstable) states of  $U$ . The energy function is analogous to an elevation function that maps a latitude and longitude coordinate in  $\mathbb{R}^2$  to an elevation in  $\mathbb{R}$ . The surface of the landscape given by the elevation map is a 2D manifold in 3D space, and the same intuition extends to higher dimensions. The energy function  $U$  defines a geodesic distance measure and non-Euclidean geometry over  $\Omega_I$  that incorporates the "elevation" information given by  $U$  to alter the usual Euclidean measure of distance on  $\mathbb{R}^N$ . This characterization of a distribution of images leads

to a geometric understanding of perceptibility based on the physical idea of *metastability* [90].

In a broad sense, a metastable system is a system that appears to be in equilibrium when viewed over short time periods but which deviates substantially from the short-scale quasi-equilibrium over long time periods. The concept of metastability provides a framework for understanding the structure of a density  $f$  by observing quasi-equilibrium behavior in the physical system associated with the landscape of  $U$ .

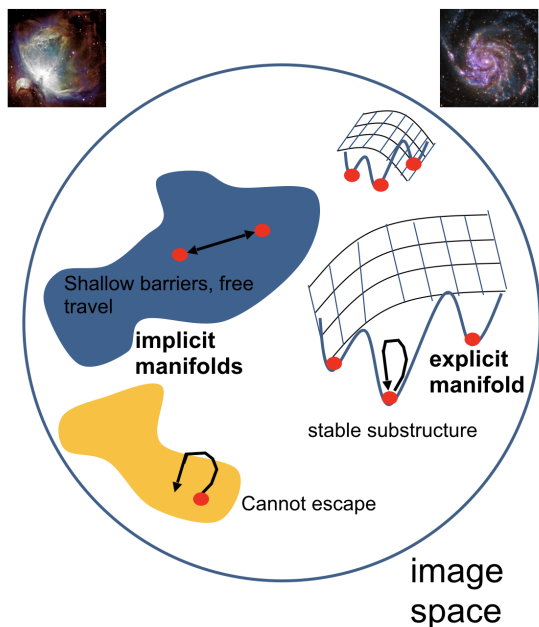


Figure 10.9: The meaningful structures of image density can be intuitively understood as high-density regions in our universe. The majority of the image universe consists of empty space that represents appearances not observed within the training set. Texton-scale images are analogous to star clusters with stable substructure that enables recognition of a distinct appearance. Texture-scale images are analogous to nebulae that whose mass covers a wide area but which contain little recognizable substructure. The gravitational pull of these structures is analogous to the metastable behavior of MCMC chains, which enables efficient mapping of macroscopic energy features.

One can computationally simulate a diffusion process in the physical system of  $U$  by obtaining MCMC samples with a steady-state  $f$ . In virtually all situations, the MCMC sampler is theoretically ergodic with respect to  $\Omega_I$ , meaning that the sampling process on  $f$  will eventually visit every image  $\mathbf{I}$  in the image space with probability 1 if the sampling is continued for a sufficient number of steps. On the other hand, it is well-known that local MCMC samplers have difficulty mixing between separate modes, which is usually considered a major drawback of MCMC methods. However, the slow mixing and high autocorrelation of local MCMC samples is actually a manifestation of metastable phenomena in the landscape of  $U$  that provide a means of understanding  $f$ .

A density  $f$  that models realistic images will have an astronomical abundance of local modes that represent the diverse variety of possible appearances along the data manifold in the image space. On the other hand, groups of related minima that represent similar images often merge in the landscape of  $U$  to form macroscopic basins or funnels that capture consistent clusters or concepts of images found in the training data. The macroscopic basins are contained regions that permit diffusion interiorly but which are separated by energy barriers that dramatically decrease the probability of cross-basin diffusion. Therefore, one can identify concepts within an image density  $f$  by identifying the metastable regions of the energy landscape. It is important to remember that metastable behavior often exists across a continuous spectrum of time-scales, so there is no ground truth for the correct metastable structure and/or conceptual clustering of  $f$ . Instead, the metastable description provides a natural way to explore the concepts within a density at a range of degrees of visual similarity, permitting "coarser" or "finer" mappings depending on the context.

We can intuitively understand the structure of the energy landscape of natural images by considering  $\Omega_I$  as a "universe" of images. The high energy/low probability regions of  $U$  are empty space, which accounts for the vast majority of the volume of  $\Omega_I$ . The low-energy regions and local modes of  $U$  represent high-density

regions. Low-energy regions that represent texon-scale images are concentrated structures like stars, while low-energy regions representing texture are more diffuse and loose structures like nebulas. Groups of related stars and nebulae form galaxies which represent general concepts within the image data, as in Figure 10.9.

The discussion above reveals an important connection between visual perceptibility of difference among a population of images with density  $f$  and metastable structures in the energy landscape  $U$ . A realistic image following  $f$  will have its own associated local minima region that captures a single stable appearance. The barriers between realistic images should depend on the degree of similarity between images. Images that share a similar appearance will be separated by lower energy barriers because it is possible to smoothly transit between similar images without encountering low probability/high energy/unrealistic images along the interpolation path. Images with dramatically different appearances should be separated by much higher barriers because it will be necessary to encounter low probability/high energy/unrealistic images while smoothly transforming between differing appearances.

For example, it is possible to smoothly transit between two images of the digit 1 while still maintaining the appearance of the digit 1 throughout the interpolation path, but it is not possible to transition from the digit 1 to the digit 0 (or any other digit) without encountering an image that does not resemble a digit at all. Groups of similar images separated by low energy barriers form metastable regions of  $U$ , establishing the connection between visual perception of differences between images and metastable structure in the energy landscape.

Metastable phenomena are a natural way of representing both structure and variation in complex concepts, but actually detecting metastable behavior in a given energy landscape is a challenging task. Intuitively, two MCMC samples initialized from the same energy basin should meet much more quickly than two MCMC samples initialized from separate energy basins. However, the "short" mixing time of two chains in the same metastable basin is still far too long for efficient simulation in high-dimensional spaces. No chains, either from the same or separate energy basins, are likely to meet in feasible time scales. Therefore direct observation of membership in a metastable basin is not possible with MCMC simulation.

To overcome the difficulty of detecting metastable phenomena in their natural state, we perturb the energy landscape in a way that will accelerate mixing within a metastable basin while preserving the long mixing-times between separate energy basins. If the perturbation is sufficiently small then it is reasonable to expect metastable phenomena in the original and altered landscape will be very similar. On the other hand, the perturbation must be strong enough to overcome shallow energy barriers that exist within a metastable basin to encourage fast mixing between modes within the same macroscopic basin.

Drawing inspiration from the magnetized Ising model, we modify the original energy landscape with an  $L_2$  penalty towards a known low-energy image. The target of the  $L_2$  penalty acts as a "representative" of the energy basin to which it belongs. Given an candidate mode  $x_0$  and a target mode  $x^*$ , initialize an MCMC sample  $X$  from  $x_0$  and update the sample using the magnetized energy

$$U_{\text{mag}}(x) = U(x) + \alpha \|x - x^*\|_2$$

where  $U$  is the original energy function and  $\alpha$  is the strength of magnetization. Sampling is continued until either the MCMC sample  $X$  comes within a small Euclidean distance  $\varepsilon$  of the target state  $x^*$  or until an upper limit on the number of steps is reached. Since  $\frac{d}{dx} \|x - x^*\|_2 = \frac{x - x^*}{\|x - x^*\|_2}$ , the gradient of the magnetization penalty  $\alpha \|x - x^*\|_2$  is a vector with magnitude  $\alpha$  pointing towards  $x^*$  for every point  $x \neq x^*$ . This shows that the gradient of  $U_{\text{mag}}$  differs from the gradient of  $U$  by a magnitude of at most  $\alpha$  throughout the state space, allowing us to uniformly control the degree of landscape perturbation in a single parameter. We call this method Attraction-Diffusion (AD).

It is clear that in the limiting case  $\alpha \rightarrow 0$ , sampling on  $U_{\text{mag}}$  is identical to sampling on  $U$  and that  $X$  will never come within a close distance of  $x^*$  even when  $x_0$  and  $x^*$  are in the same metastable basin. On the other

hand, in the limiting case  $\alpha \rightarrow \infty$  all probability mass is focused on  $x^*$  and  $X$  will quickly reach the target state. In between these limiting cases there exists a spectrum of values for  $\alpha$  for which the gradients from the original energy  $U$  and the magnetization penalty  $\alpha\|x - x^*\|_2$  affect sampling at approximately the same magnitude. Within this spectrum, one observes that sometimes the diffusion paths reach their target  $x^*$  and sometimes the diffusion paths never reach their target. If we consider successful travel in the magnetized landscape to approximately represent metastable membership in the original landscape, then it becomes possible to reason about metastable structures in the original landscape based on the finite-step behavior of MCMC samples in the magnetized landscape. We note that metastable structures at finer resolutions can be detected when  $\alpha$  is relatively small, while larger values of  $\alpha$  will only preserve the most prominent barriers in the landscape. Like real-world concepts, metastable concepts only exist within a certain spectrum of identity or difference depending on the context of the situation.

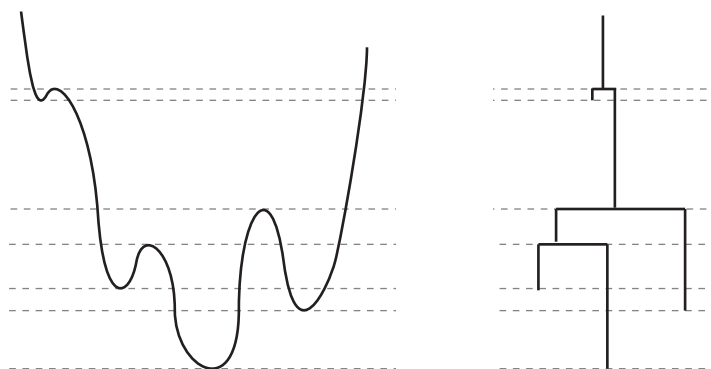


Figure 10.10: Reprinted with permission from [90]. Construction of a Disconnectivity Graph (right) from a 1D energy landscape (left). The DG encodes the depth of local minima and the energy at which basins merge. The same procedure can be used to hierarchically cluster image concepts using an energy function  $U$  over the image space.

Now that we are equipped with a method for determining whether two minima belong to the same metastable basin, it becomes possible to efficiently identify the different metastable structures in an arbitrary high-dimensional energy landscape using computational methods. This process allows to extend techniques for mapping and visualizing energy landscapes from the chemical physics literature to energy functions defined over the image space. The physical system defined by a low-temperature diffusion process has metastable structures that corresponds to concepts learned by the energy function.

The analogy between stable states of a physical system (e.g. states with predominantly aligned spins in the Ising model) and recognizable concepts within groups of images allows us to hierarchically cluster the image space by first identifying metastable regions then grouping regions based on the energy spectrum at which basins merge. The results can be displayed in a tree diagram known as a Disconnectivity Graph (DG). A Disconnectivity Graph displays 1) the minimum energy within each basin (leaves of the tree), 2) the energy level at which basins merge in the energy landscape, also called the energy barrier (branches of the tree), as shown in Figure 10.10. The key concept underlying this procedure is the idea that perceptibility can be grounded in metastable phenomena in an energy landscape that represents perceptual memory.

We demonstrate the principles of the above discussion by mapping a learned energy landscape  $U$  that has been trained to model MNIST. It is important to note that the true density of an image concept is never known and that mapping must be done on a *learned* density.  $U$  is a computational representation of a large set of observed images, much like our memory creates perceptual models of structured images that

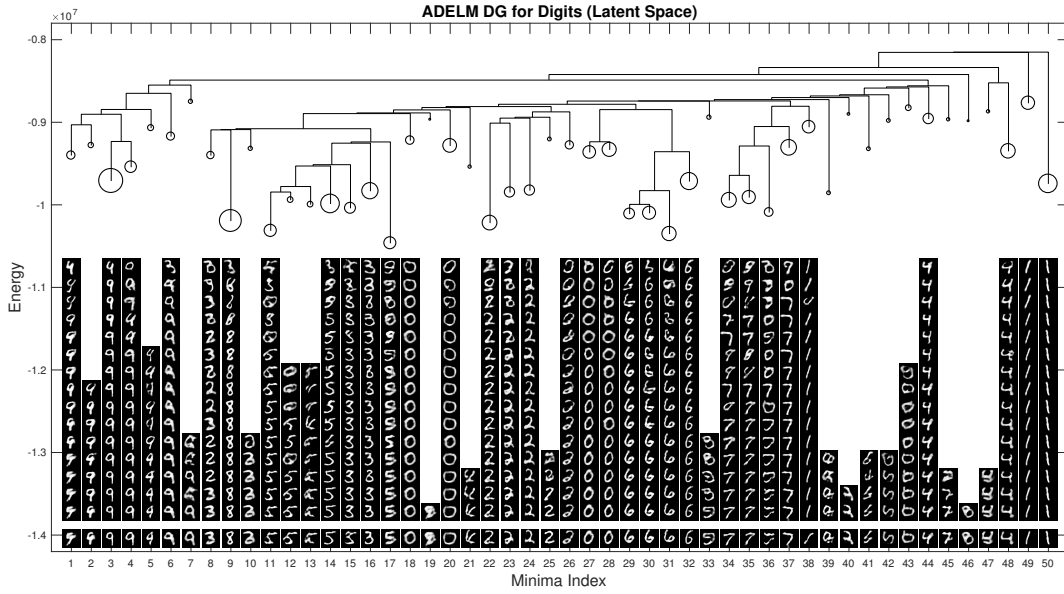


Figure 10.11: Reprinted with permission from [90]. Map of the energy landscape of an image density trained with image patches from an ivy texture at four scales. The closest two scales develop recognizable and separate basins in the landscape that represent distinguishable patterns. The furthest two scales do not contain perceptible subgroups and form flat nebula-like basins that encode texture appearance.

we regularly observe. The macroscopic energy structures of  $U$  can effectively distinguish between different image groups (digits) that are recognizable to humans. Each digit has at least one stable basin that represents the digit appearance, and relations between image basins follow visually intuitive relations between images.

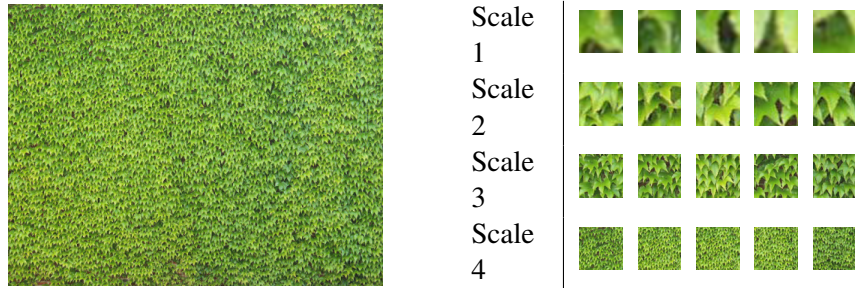


Figure 10.12: Reprinted with permission from [90]. Ivy texture image and image patches from four scales. The first two scales contain images which can be clustered by a human. The images cross the perceptibility threshold from texton representation to texture representation between Scale 2 and Scale 3.

The link between perceptibility and metastability can also be clearly observed when mapping the density of image patches from a single texture at a variety of different scales. The density  $f$  models  $32 \times 32$  pixel image patches of the same ivy texture image from four different scales (see Figure 10.12). At the closest scale, the images are composed of simple bar and stripe features. The second-closest scale features the composition of about 2 or 3 leaves in different arrangements. At the furthest two scales, it is difficult to distinguish distinct image groups and images from these scales are perceived as textures by a human.

The metastable structure of  $U = -\log f$  shares many similarities with human perceptibility (see Figure 10.13). Image groups from the closest scale are easiest to recognize and the landscape forms a handful

of strong basins that represent the different bar and stripe configurations found at close range. A rich variety of separate metastable basins appear at Scale 2 to encode many distinct composition of a few leaves. The landscape represents images from Scales 3 and 4 with a large macroscopic basin with little substructure. Between Scale 2 and Scale 3, a phase transition occurs where the identifiability of leaf compositions changes from distinguishable (texton scale) to indistinguishable (texture scale). The behavior of human perceptibility and image landscape metastability are therefore quite similar across multi-scale image data.

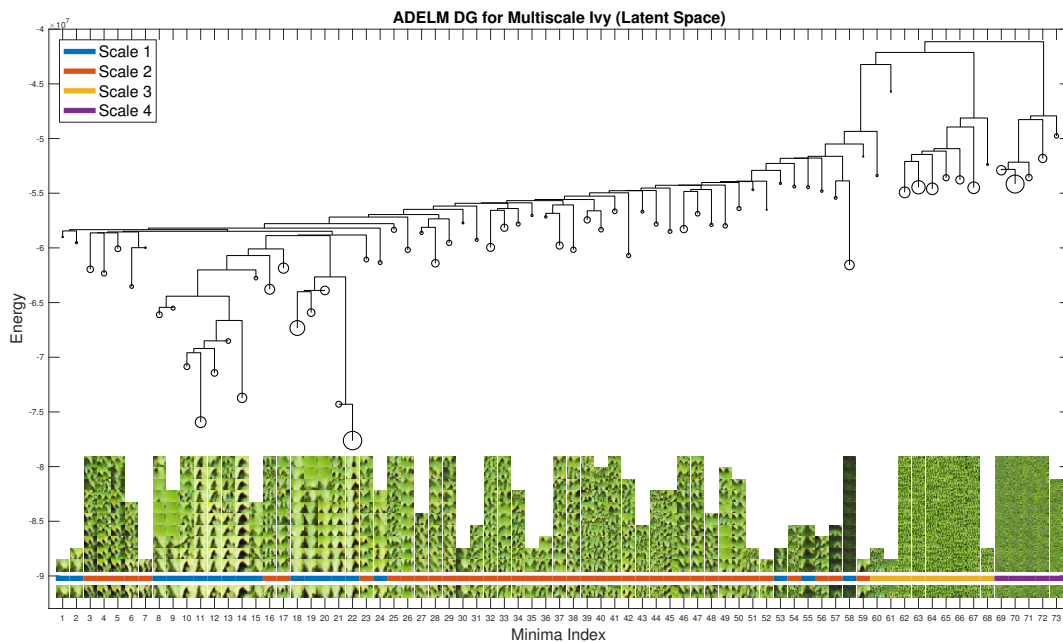
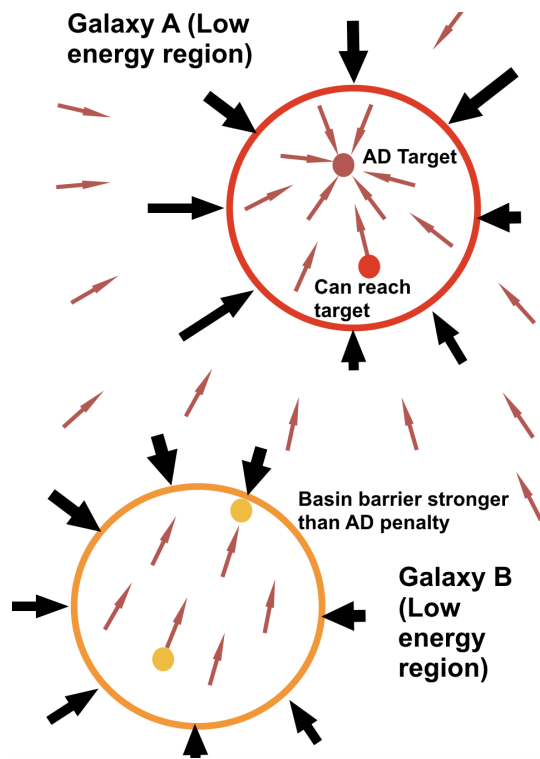


Figure 10.13: Reprinted with permission from [90]. Map of the energy landscape of an image density trained with image patches from an ivy texture at four scales. The closest two scales develop recognizable and separate basins in the landscape that represent distinguishable patterns. The furthest two scales do not contain perceptible subgroups and form flat nebula-like basins that encode texture appearance.







# 11

## Deep Image Models

In earlier chapters, we followed Marr’s traditional vision and desired to integrate high-entropy (e.g. texture) and low-entropy (e.g. textons) regimes into primal sketch models. It appears that Markov random fields in the form of the FRAME model are preferable for high-entropy regimes whereas a sparse-coding model is limited to the low-entropy regime. However, with advances in deep convolutional networks (ConvNets), we may generalize both the FRAME model and sparse coding to cover the entire spectrum of entropy.

In this chapter, we will present the DeepFRAME model as a recursive multi-layer generalization of the original FRAME model, and, the Generator model which can be considered a non-linear multi-layer generalization of the factor analysis model. Such multi-layer models capture the fact that visual patterns and concepts appear at multiple layers of abstractions.

### 11.1 Deep FRAME

The original FRAME (Filters, Random fields, And Maximum Entropy) model [237, 261, 262] is a Markov random field model of stationary spatial processes such as stochastic textures. The log probability density function of the model is the sum of translation-invariant potential functions that are point-wise one-dimensional non-linear transformations of linear filter responses. It specifies a probability distribution on the signal  $\mathbf{I}$  via the unnormalized log density function that is a linear combination of the features,

$$p_{\theta}(\mathbf{I}) = \frac{1}{Z(\theta)} \exp \left\{ h(\mathbf{I})^{\top} \theta \right\} p_0(\mathbf{I}), \quad (11.1)$$

where  $h(\mathbf{I})$  is the  $d$ -dimensional feature vector extracted from  $\mathbf{I}$ , and  $\theta$  is the  $d$ -dimensional vector of weight parameters.  $p_0(\mathbf{I})$  is a known reference distribution such as the white noise model  $\mathbf{I} \sim \mathcal{N}(0, \sigma^2 \mathbf{I}_p)$ , or the uniform distribution within a bounded range.

$$Z(\theta) = \int \exp \{ h(\mathbf{I})^{\top} \theta \} p_0(\mathbf{I}) d\mathbf{I} = \mathcal{E}_{p_0} [\exp \{ h(\mathbf{I})^{\top} \theta \}] \quad (11.2)$$

is the normalizing constant ( $\mathcal{E}_p$  denotes the expectation with respect to  $p$ ), which is analytically intractable on high dimensional space.

In the FRAME model,  $h(\mathbf{I})$  consists of histograms of responses from a bank of filters. In a simplified non-convolutional version,  $h(\mathbf{I})^{\top} \theta = g(W\mathbf{I}) = \sum_{k=1}^d g_k(W_k \mathbf{I})$ , where  $W$  is a  $d \times p$  matrix, and  $W_k$  is the  $k$ -th row of  $W$ .  $W\mathbf{I}$  consists of the  $d$  filter responses with each row of  $W$  being a linear filter. ( $g_k, k = 1, \dots, d$ ) are  $d$  one-dimensional potential functions applied respectively to the  $d$  elements of  $W\mathbf{I}$ . In the FRAME model, the rows of  $W$  are a bank of Gabor wavelets or filters [34]. The input value of each

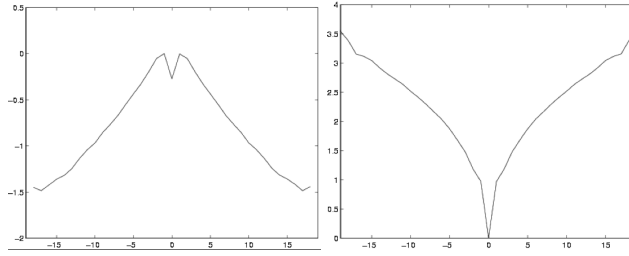


Figure 11.1: Two types of potential functions learned by [259] from natural images. The function on the left encourages big filter responses and creates patterns via reaction, while the function on the right prefers small filter responses and smoothes out the synthesized image via diffusion.

potential function  $g_k$  is discretized to several bins and the output value within each bin is assumed to be a constant. Given the predefined filters, [259] learned the potential functions  $(g_k, k = 1, \dots, d)$  from natural images. There are two types of potential functions as shown in Figure 11.1 taken from [259]. The function on the left encourages big filter responses while the function on the right prefers small filter responses. [259] used the Langevin dynamics to sample from the learned model. The gradient descent component of the dynamics is interpreted as the Gibbs Reaction And Diffusion Equations (GRADE), where the function on the left of Figure 11.1 is for reaction to create patterns, while the function on the right is for diffusion to smooth out the synthesized image. Figure 11.2 shows the learning process of a two dimensional FRAME model by sequentially adding filters, i.e., adding rows to  $W$ . Each filter corresponds to a projection of the data along a certain direction. Each learning step finds the projection that reveals the maximum difference between the observed data and the synthesized data by the current model, and matches the statistics of the observed data and synthesized data along that direction. Such a learning method is also called matching pursuit.

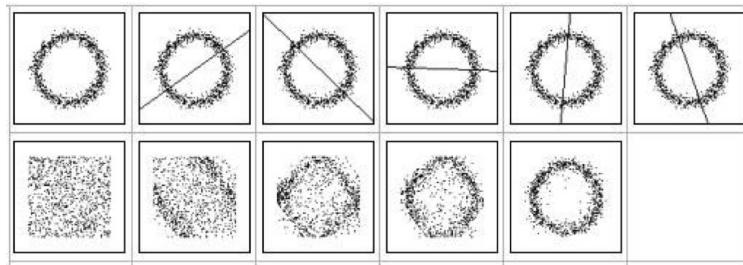


Figure 11.2: Learning a two dimensional FRAME model by sequentially adding rows to  $W$  [138]. Each row of  $W$  corresponds to a projection of the data. Each step finds the projection that reveals the maximum difference between the observed data and the synthesized data generated by the current model.

The deep FRAME generalization of the FRAME model is inspired by the successes of deep convolutional neural network [124, 127], and as the name suggests, it can be considered a deep version of the FRAME model. The filters used in the original FRAME model are linear filters that capture local image features. In [146], the linear filters are replaced by the non-linear filters at a certain convolutional layer of a pre-trained deep ConvNet. Such filters can capture more complex patterns, and the deep FRAME model built on such filters can be more expressive. Instead of using filters from a pre-trained ConvNet, we can also learn the filters from the observed data. The resulting model is also known as a deep convolutional

energy-based model [31, 165, 244]. Such a model can be considered a recursive multi-layer generalization of the original FRAME model. The log probability density function of the original FRAME model consists of non-linear transformations of linear filter responses. If we repeat this structure recursively, we get the generative ConvNet model with multiple layers of linear filtering followed by point-wise non-linear transformations. [244] shows that it is possible to learn such a model from natural images.

The deep FRAME model can be written as a general exponential tilting of a reference distribution such as the uniform measure or the Gaussian white noise model. If the reference distribution is the Gaussian white noise model, the local modes of the probability density follow an auto-encoder. We call it the Hopfield auto-encoder, because it defines the local energy minima of the model [97]. In the Hopfield auto-encoder, the bottom-up filters detect the patterns corresponding to the filters, then the binary detection results are used as the coefficients in the top-down representation where the original filters play the role of basis functions.

The learning of deep FRAME model follows an analysis-by-synthesis scheme [77]. We can use Markov chain Monte Carlo (MCMC) such as the Langevin dynamics to sample from the current model to generate synthetic images, which runs gradient descent on the energy function of the model while adding Gaussian white noises. For deep FRAME models, the gradient can be efficiently computed by back-propagation. Then we update the model parameters based on the statistical difference between the observed images and the synthetic images, so that the model shifts its probability density function, especially the high-density regions or the low energy regions, from the synthetic images to the observed images. In the zero-temperature limit, this learning and sampling algorithm admits an adversarial interpretation, where the learning step and the sampling step play a minimax game based on a value function. For sampling synthetic images, a generator model [74, 121, 159, 185] can be recruited as a much more efficient non-iterative approximate sampler that replaces the MCMC sampling by direct ancestral sampling.

### 11.1.1 From FRAME to Deep FRAME

In this section, we use toy 2D examples to illustrate the evolution from the FRAME model to the deep FRAME model.

Recall that in the original FRAME model, the unnormalized log density function of input signal  $\mathbf{I}$  is parametrized as  $h(\mathbf{I})^\top \theta = g(W\mathbf{I}) = \sum_{k=1}^d g_k(W_k\mathbf{I})$ , where  $(g_k, k = 1, \dots, d)$  are  $d$  one-dimensional learnable potential functions. We can slightly modify the parametrization as

$$h(\mathbf{I})^\top \theta = \text{ReLU}(W\mathbf{I})^\top \theta = \max(0, W\mathbf{I})^\top \theta, \quad (11.3)$$

which can be considered a two-layer feedforward neural network. The ReLU function operates in a point-wise fashion and clips negative values to zero. Given a filter (i.e., a row of  $W$ ), the original FRAME model learns to match the histograms of the projection by the filter of the observed data and the synthetic data (Figure 11.3a), whereas the modified parametrization matches the expected value of  $\text{ReLU}(W\mathbf{I})$  (Figure 11.3b).

Figure 11.4 demonstrates the matching pursuit process of the modified FRAME model, learned on a 2D distribution consisting of two mixtures of Gaussian distributions. Each filter partition the space into two regions by a straight line, one with positive filter responses while the other one with filter responses clipped to zero. The space is partitioned into small pieces by all filters. Within each piece, the activation patterns of all filters are the same and the model is piecewise linear. By adding more filters, the model is able to depict more complex nonlinear functions with the piecewise linear approximation.

We can further modify the parametrization of the model to a deeper neural network and learn the filters at multiple layers using the gradient descent algorithm, which leads to the deep FRAME model. Take a three-layer feedforward neural network with ReLU activations as an example. As illustrated in Figure 11.3c, the

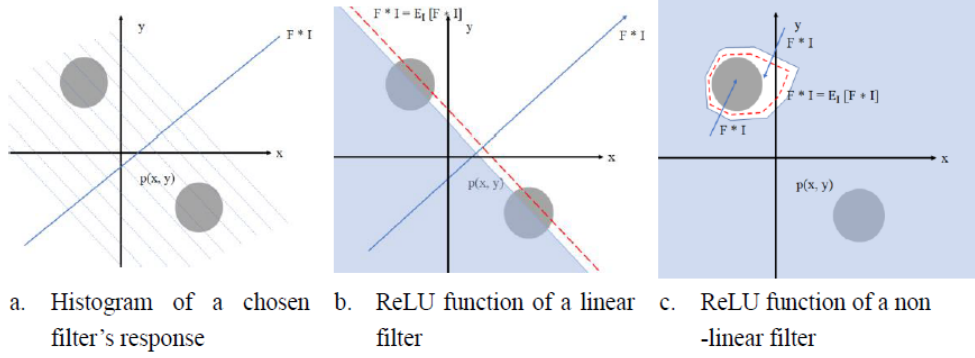


Figure 11.3: Suppose the 2D observed data locates in two discrete modes (grey circles). A linear filter corresponds to a projection of the data (the blue lines in (a) and (b)). (a) The original FRAME model matches the histogram of projection of the observed data and synthesized data. (b) The modified FRAME model with ReLU activation. The ReLU function clips the negative filter responses (shadow area) to zero. Such a model matches the expected value of the clipped filter responses of the observed data and synthesized data. We visualize the points whose filter responses are equal to the expected value (i.e., the contour line at the expected value) as the red dotted line. (c) A three-layer deep FRAME model matches the expected response of a non-linear filter defined by the bottom two layers. The contour line and the area with positive filter responses (the unshaded area) are more complex and they are closer to one of the data modes.

filter at the second layer is nonlinear, which partitions the space in a more complex manner than a linear filter. Figure 11.5 shows the learned results of the deep Frame model parametrized by a three-layer neural network. The network has seven filters in the first layer and three filters in the second one. Compared to its shallower counterpart (Figure 11.4), the model achieves more accurate density estimation. More interestingly, the two filters in the second layer identify the two discrete modes of the observed data distribution respectively. In this particular case, we may even be able to cluster the observed data by the activation patterns of the two filters.

The advantage of a deeper model is more distinct when the data distribution is more complicated. Figure 11.6 demonstrates the density estimation results for an irregular data distribution. A three-layer deep FRAME model is about to estimate the density reasonably well with 24 filters in total, while a two-layer model performs poorly with a double amount of filters.

### 11.1.2 ConvNet Filters

The convolutional neural network (CNN or ConvNet) [127] is a specialized neural network devised for analyzing signals such as images, where the linear transformations take place around each pixel, i.e., they are filters or convolutions. See Figure 11.7 for an illustration.

A ConvNet consists of multiple layers of linear filtering and point-wise non-linear transformation, as expressed by the following recursive formula:

$$[F_j^{(l)} * \mathbf{I}](y) = h \left( \sum_{k=1}^{N_{l-1}} \sum_{x \in \mathbb{S}_l} w_{k,x}^{(l,j)} [F_k^{(l-1)} * \mathbf{I}](y+x) + b_{l,j} \right), \quad (11.4)$$

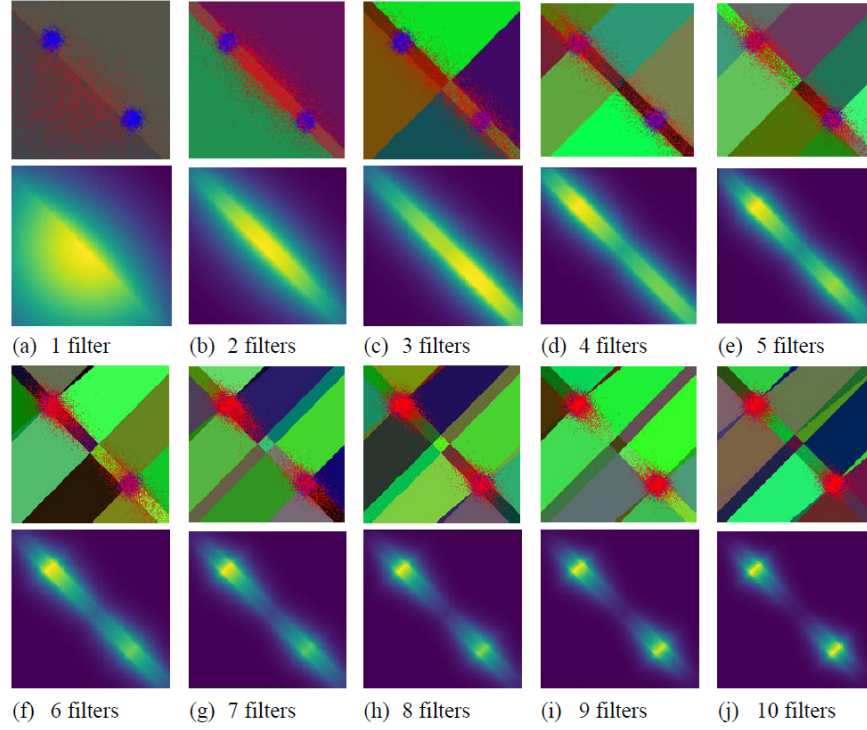


Figure 11.4: Matching pursuit of the modified FRAME model with ReLU activations. The filters partition the space into small pieces. Within each piece, the activation patterns of all filters are the same, so that the model is piecewise linear. For each subfigure, the top figure shows the true data distribution (blue points), the partition of the space (in the background), and samples drawn from the current distribution (red points), while the bottom figure shows the estimated density function.

or

$$\mathbf{I}_j^{(l)}(y) = h \left( \sum_{k=1}^{N_{l-1}} \sum_{x \in \mathcal{S}_l} w_{k,x}^{(l,j)} \mathbf{I}_k^{(l-1)}(y+x) + b_{l,j} \right), \quad (11.5)$$

where  $l = 1, \dots, L$  indexes the layer, and  $\mathbf{I}_j^{(l)} = F_j^{(l)} * \mathbf{I}$  are filtered images or feature maps at layer  $l$ . In Figure 11.7, the feature maps are illustrated by the square shapes. Each  $[F_j^{(l)} * \mathbf{I}](x)$  is called a filter response or a feature extracted by a node or a unit at layer  $l$ .

$\{F_j^{(l)}, j = 1, \dots, N_l\}$  are the filters at layer  $l$ , and  $\{F_k^{(l-1)}, k = 1, \dots, N_{l-1}\}$  are the filters at layer  $l-1$ .  $j$  and  $k$  are used to index the filters at layers  $l$  and  $l-1$  respectively, and  $N_l$  and  $N_{l-1}$  are the numbers of filters at layers  $l$  and  $l-1$  respectively. The filters are locally supported, so the range of  $\mathbf{I}$  in  $\sum_x$  is within a local support  $\mathcal{S}_l$  (such as a  $7 \times 7$  image patch). We let  $\mathbf{I}^{(0)} = \mathbf{I}$ . The filter responses at layer  $l$  are computed from the filter responses at layer  $l-1$ , by linear filtering defined by the weights  $w_{k,x}^{(l,j)}$  as well as the bias term  $b_{l,j}$ , followed by the non-linear transformation  $h(\cdot)$ . The mostly commonly used non-linear transformation in the modern ConvNets is the rectified linear unit (ReLU),  $h(r) = \max(0, r)$  [124].  $\{F_j^{(l)}\}$  are non-linear filters because we incorporate  $h(\cdot)$  in the computation of the filter responses. We call  $\mathbf{I}_j^{(l)} = F_j^{(l)} * \mathbf{I}$  the filtered image or the feature map of filter  $j$  at layer  $l$ . We denote  $\mathbf{I}^{(l)} = (\mathbf{I}_j^{(l)}, j = 1, \dots, N_l)$ , which consists of a total of  $N_l$  feature maps at layer  $l$ , and  $j = 1, \dots, N_l$ . Sometimes, people call  $\mathbf{I}^{(l)}$  as a whole feature map

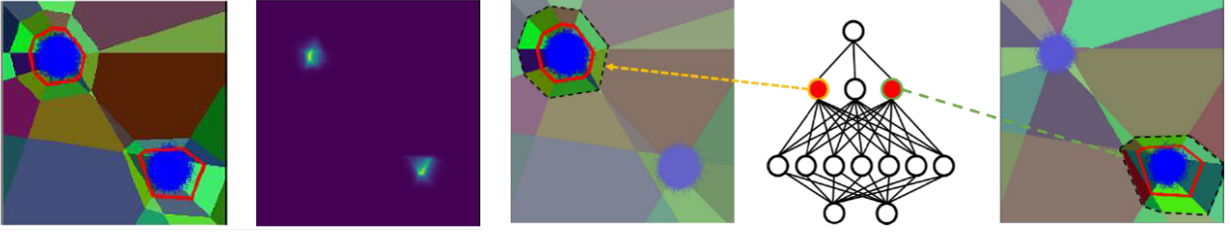


Figure 11.5: Learned results of a Deep FRAME model parametrized by a three-layer feedforward neural network. *Left*: observed data (blue points), partitions of the space by the filters and contour line at the expected value of the filter responses of the observed data (red line). *Middle*: estimated density function. *Right*: areas with positive responses of two filters in the second layer, which identify the two discrete modes of the data distribution respectively.

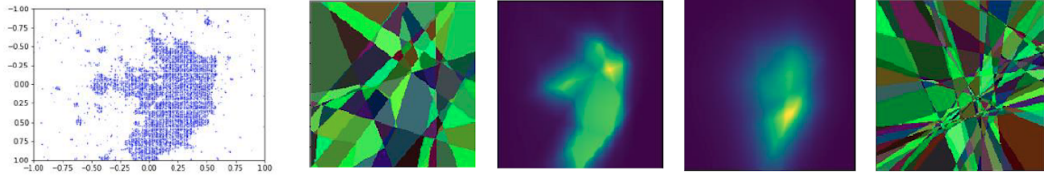


Figure 11.6: Density estimation results for an irregular data distribution. *Left*: the data distribution. *Middle*: the results of a three-layer model learned by gradient descent (16 neurons in the first layer and 8 neurons in the second layer). *Right*: the results of a two-layer model learned by matching pursuit (48 neurons in total)

or filter image with  $N_l$  channels, where each  $\mathbf{I}_j^{(l)}$  corresponds to one channel. For a colored image,  $\mathbf{I}^{(0)} = \mathbf{I}$  has 3 channels for RGB.

The filtering operations are often followed by sub-sampling and local-max pooling (e.g.,  $\mathbf{I}(x_1, x_2) \leftarrow \max_{(s_1, s_2) \in \{0,1\}^2} \mathbf{I}(2x_1 + s_1, 2x_2 + s_2)$ ). See Figure 11.7 for an illustration of sub-sampling. After a number of layers with sub-sampling, the filtered images or feature maps are reduced to  $1 \times 1$  at the top layer. These features are then used for classification (e.g., does the image contain a hummingbird or a seagull or a dog) via multinomial logistic regression.

### 11.1.3 FRAME with ConvNet Filters

Instead of using linear filters as in the original FRAME model, we can use the filters at a certain convolutional layer of a pre-learned ConvNet. We call such a model the deep FRAME model.

Suppose there exists a bank of filters  $\{F_k^{(l)}, k = 1, \dots, K\}$  at a certain convolutional layer  $l$  of a pre-learned ConvNet, as recursively defined by (11.4). For an image  $\mathbf{I}$  defined on the image domain  $\mathcal{D}$ , let  $F_k^{(l)} * \mathbf{I}$  be the feature map of filter  $F_k^{(l)}$ , and let  $[F_k^{(l)} * \mathbf{I}](x)$  be the filter response of  $\mathbf{I}$  to  $F_k^{(l)}$  at position  $\mathbf{I}$  (again  $\mathbf{I}$  is a two-dimensional coordinate). We assume that  $[F_k^{(l)} * \mathbf{I}](x)$  is the response obtained after applying the non-linear transformation or rectification function  $h(\cdot)$ . Then the non-stationary deep FRAME model becomes

$$p(\mathbf{I}; \theta) = \frac{1}{Z(\theta)} \exp \left\{ \sum_{k=1}^K \sum_{x \in \mathcal{D}} w_{k,x} [F_k^{(l)} * \mathbf{I}](x) \right\} q(\mathbf{I}), \quad (11.6)$$

where  $q(\mathbf{I})$  is again the Gaussian white noise model (4.19), and  $w = (w_{k,x}, \forall k, x)$  are the unknown param-

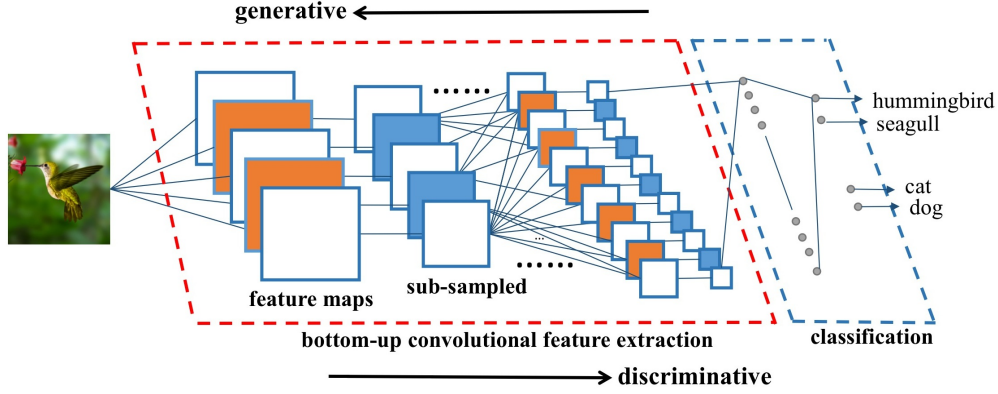


Figure 11.7: Reprinted with permission from [145]. Convolutional neural networks consist of multiple layers of filtering and sub-sampling operations for bottom-up feature extraction, resulting in multiple layers of feature maps and their sub-sampled versions. The top layer features are used for classification via multinomial logistic regression. The discriminative direction is from image to category, whereas the generative direction is from category to image.

eters to be learned from the training data. Model (11.6) shares the same form as model (4.18) with linear filters, except that the rectification function  $h(r) = \max(0, r)$  in model (4.18) is already absorbed in the ConvNet filters  $\{F_k^{(l)}\}$  in model (11.6). We can also make model (11.6) stationary by letting  $w_{k,x} = w_k$  for all  $\mathbf{I}$ .

#### 11.1.4 Learning and Sampling

The basic learning algorithm estimates the unknown parameters  $w$  from a set of aligned training images  $\{\mathbf{I}_i, i = 1, \dots, n\}$  that come from the same object category. Again the weight parameters  $w$  can be estimated by maximizing the log-likelihood function, and  $w$  can be computed by the stochastic gradient ascent algorithm [254]:

$$w_{k,x}^{(t+1)} = w_{k,x}^{(t)} + \gamma_t \left[ \frac{1}{n} \sum_{i=1}^n [F_k^{(l)} * \mathbf{I}_i](x) - \frac{1}{\tilde{n}} \sum_{i=1}^{\tilde{n}} [F_k^{(l)} * \tilde{\mathbf{I}}_i](x) \right] \quad (11.7)$$

for every  $k \in \{1, \dots, K\}$  and  $x \in \mathcal{D}$ , where  $\gamma_t$  is the learning rate, and  $\{\tilde{\mathbf{I}}_i, i = 1, \dots, \tilde{n}\}$  are the synthetic images sampled from  $p(\mathbf{I}; w^{(t)})$  using MCMC. This is an analysis by synthesis scheme that seeks to match the average filter responses of the synthetic images to those of the observed images.

In order to sample from  $p(\mathbf{I}; \theta)$ , we adopt the Langevin dynamics [73, 141]. Writing the energy function

$$U(\mathbf{I}, \theta) = - \sum_{k=1}^K \sum_{x \in \mathcal{D}} w_{k,x} [F_k^{(l)} * \mathbf{I}](x) + \frac{1}{2\sigma^2} \|\mathbf{I}\|^2, \quad (11.8)$$

the Langevin dynamics iterates

$$\mathbf{I}_{\tau+1} = \mathbf{I}_{\tau} - \delta U'(\mathbf{I}_{\tau}, \theta) + \sqrt{2\delta} \epsilon_{\tau}, \quad (11.9)$$

where  $U'(\mathbf{I}, \theta) = \partial U(\mathbf{I}, \theta) / \partial \mathbf{I}$ . This gradient can be computed by back-propagation. In (12.76),  $\delta$  is a small step-size, and  $\epsilon_{\tau} \sim N(0, I_D)$ , independently across  $\tau$ , where  $I_D$  is the identity matrix of dimension



$D = |\mathcal{D}|$ , i.e., the dimensionality of  $\mathbf{I}$ .  $\epsilon_\tau$  is a Gaussian white noise image whose pixel values follow  $N(0, 1)$  independently. Here we use  $\tau$  to denote the time steps of the Langevin sampling process, because  $t$  is used for the time steps of the learning process. The Langevin sampling process (12.76) is an inner loop within the learning process (11.7). Between every two consecutive updates of  $w$  in the learning process, we run a finite number of steps of the Langevin dynamics starting from the images generated by the previous iteration of the learning algorithm.

The Langevin dynamics was first applied to the FRAME model by [261], where the gradient descent component is interpreted as the Gibbs Reaction And Diffusion Equation (GRADE), and the patterns are formed via the reactions and diffusions controlled by different types of filters.

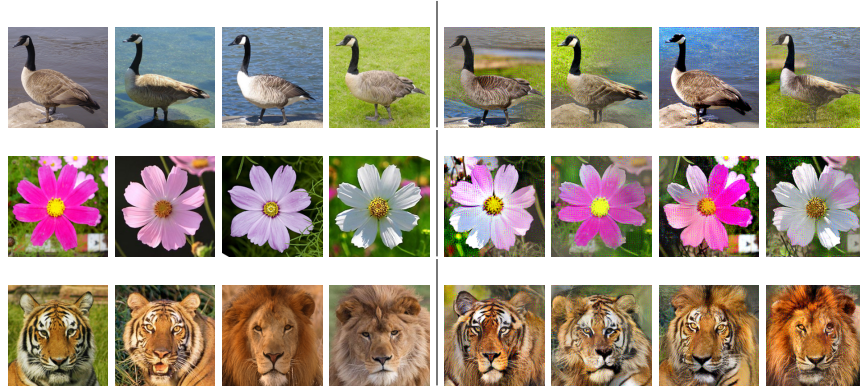


Figure 11.8: Reprinted with permission from [146]. Generating object patterns. In each row, the left half displays 4 of the training images ( $224 \times 224$ ), and the right half displays 4 of the synthetic images. In the last row, the learned model generates hybrid patterns of lion and tiger.

We first learn a non-stationary FRAME model (11.6) from images of aligned objects. The images were collected from the internet. For each category, the number of training images was around 10. We used  $\tilde{n} = 16$  parallel chains for Langevin sampling with 100 Langevin steps between every two consecutive updates of the parameters. Figure 11.8 shows some experiments using filters from the 3rd convolutional layer of the VGG ConvNet [202], a commonly used pre-learned ConvNet trained on Imagenet ILSVRC2012 dataset [37]. For each experiment on each row, the left half displays 4 of the training images, and the right half displays 4 of the synthetic images generated by the Langevin dynamics. The last experiment is about learning the hybrid pattern of lion and tiger. The model re-mixes local image patterns seamlessly.

Figure 11.9 shows results from experiments on the stationary model for texture images. The model does not require image alignment. It re-shuffles the local patterns seamlessly. Each experiment is illustrated by 3 images, where the first image is the training image, and the other 2 images are generated by the learning algorithm. In the last 3 images, the first 2 images are training images, and the last image is generated by the learned model that mixes the patterns of brick wall and ivy.

### 11.1.5 Learning a New Layer of Filters

On top of the existing pre-learned convolutional layer of filters  $\{F_k^{(l)}, k = 1, \dots, K\}$ , we can build another layer of filters  $\{F_j^{(l+1)}, j = 1, \dots, J\}$ , according to the recursive formula (11.4), so that

$$[F_j^{(l+1)} * \mathbf{I}](y) = h \left( \sum_{k,x} w_{k,x}^{(j)} [F_k^{(l)} * \mathbf{I}](y+x) + b_j \right), \quad (11.10)$$



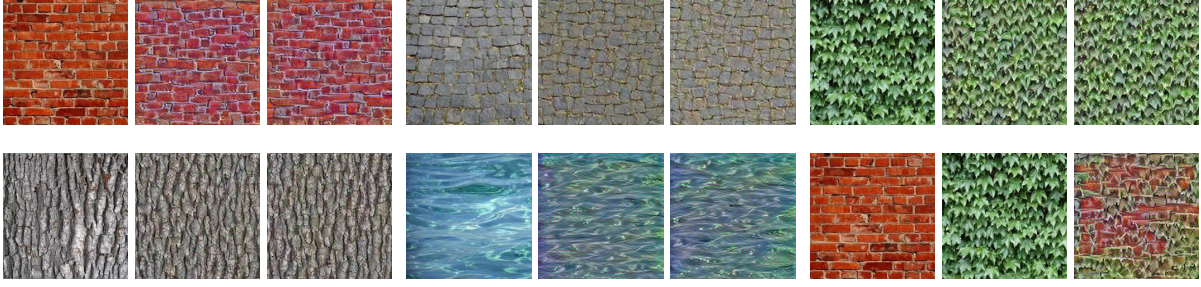


Figure 11.9: Reprinted with permission from [146]. Generating texture patterns. For each category, the first image ( $224 \times 224$ ) is the training image, and the next 2 images are generated images, except for the last 3 images, where the first 2 are the training images, and the last one is the generated image that mixes brick wall and ivy.

where  $h(r) = \max(0, r)$ . The set  $\{F_j^{(l+1)}\}$  is like a dictionary of “words” to describe different types of objects or patterns in the training images.

Due to the recursive nature of ConvNet, the deep FRAME model (11.6) based on filters  $\{F_k^{(l)}\}$  corresponds to a single filter in  $\{F_j^{(l+1)}\}$  at a particular position  $y$  (e.g., the origin  $y = 0$ ) where we assume that the object appears. In [31], we show that the rectification function  $h(r) = \max(0, r)$  can be justified by a mixture model where the object can either appear at a position or not. The bias term is related to  $-\log Z(\theta)$ .

Model (11.6) is used to model images where the objects are aligned and are of the same category. For images of non-aligned objects from multiple categories, we can extend model (11.6) to a convolutional version with a whole new layer of multiple filters

$$p(\mathbf{I}; \theta) = \frac{1}{Z(\theta)} \exp \left\{ \sum_{j=1}^J \sum_{x \in \mathcal{D}} [F_j^{(l+1)} * \mathbf{I}](x) \right\} q(\mathbf{I}), \quad (11.11)$$

where  $\{F_j^{(l+1)}\}$  are defined by (11.10). This model is a product of experts model [91, 188], where each  $[F_j^{(l+1)} * \mathbf{I}](x)$  is an expert about a mixture of an activation or inactivation of an object of type  $j$  at position  $\mathbf{I}$ . The stationary model for textures is a special case of this model.

Suppose we observe images of non-aligned objects from multiple categories  $\{\mathbf{I}_i, i = 1, \dots, n\}$ , and we want to learn a new layer of filters  $\{F_j^{(l+1)}, j = 1, \dots, J\}$  by fitting the model (11.11) with (11.10) to the observed images, where  $\{F_j^{(l+1)}\}$  model different types of objects in these images. This is an unsupervised learning problem because we do not know where the objects are. The model can still be learned by the analysis by synthesis scheme as before.

Let  $L(\theta) = \frac{1}{n} \sum_{i=1}^n \log p(\mathbf{I}_i; \theta)$  be the log-likelihood where  $p(\mathbf{I}; \theta)$  is defined by (11.11) and (11.10). Then the gradient ascent learning algorithm is based on

$$\frac{\partial L(\theta)}{\partial w_{k,x}^{(j)}} = \frac{1}{n} \sum_{i=1}^n \sum_{y \in \mathcal{D}} s_{j,y}(\mathbf{I}_i) [F_k^{(l)} * \mathbf{I}_i](y+x) - \mathcal{E}_w \left[ \sum_{y \in \mathcal{D}} s_{j,y}(\mathbf{I}) [F_k^{(l)} * \mathbf{I}](y+x) \right], \quad (11.12)$$

where

$$s_{j,y}(\mathbf{I}) = h' \left( \sum_{k,x} w_{k,x}^{(j)} [F_k^{(l)} * \mathbf{I}](y+x) + b_j \right) \quad (11.13)$$

is a binary on/off detector of object  $j$  at position  $y$  on image  $\mathbf{I}$ , because for  $h(r) = \max(0, r)$ ,  $h'(r) = 0$  if  $r \leq 0$ , and  $h'(r) = 1$  if  $r > 0$ . The gradient (11.12) admits an EM [36] interpretation which is typical in unsupervised learning algorithms that involve hidden variables. Specifically,  $s_{j,y}()$  detects the object of type  $j$  that is modeled by  $F_j^{(l+1)}$  at location  $y$ . This step can be considered a hard-decision E-step. With the objects detected, the parameters of  $F_j^{(l+1)}$  are then refined in a similar way as in (11.7), which can be considered the M-step. That is, we learn  $F_j^{(l+1)}$  only from image patches where objects of type  $j$  are detected.



Figure 11.10: Reprinted with permission from [146]. Learning a new layer of filters without requiring object bounding boxes or image alignment. For each experiment, the first image ( $224 \times 224$ ) is the training image, and the next 2 images are generated by the learned model.

Figure 11.10 displays two experiments. In each experiment, the first image ( $224 \times 224$ ) is the training image, and the rest 2 images are generated by the learned model. In the first scenery experiment, we learn 10 filters at the 4th convolutional layer, based on the pre-trained VGG filters at the 3rd layer. The size of each Conv4 filter to be learned is  $11 \times 11 \times 256$ . In the second sunflower experiment, we learn 20 filters of size  $7 \times 7 \times 256$ . Clearly, these learned filters capture the local objects or patterns and re-shuffle them seamlessly.

### 11.1.6 Deep Convolutional Energy-Based Model

Instead of relying on the pre-trained filters from an existing ConvNet, we can also learn the filters  $\{F_k^{(l)}, k = 1, \dots, K\}$  themselves. The resulting model is a deep convolutional energy-based model (EBM) [31, 165, 244],

$$p(\mathbf{I}; \theta) = \frac{1}{Z(\theta)} \exp\{f(\mathbf{I}; \theta)\} q(\mathbf{I}), \quad (11.14)$$

where  $f(\mathbf{I}; \theta)$  is defined by a ConvNet. In model (11.11) with (11.10), we have

$$f(\mathbf{I}; \theta) = \sum_{j=1}^J \sum_{x \in \mathcal{D}} [F_j^{(l+1)} * \mathbf{I}](x). \quad (11.15)$$

Using more compact notation, we can define  $f(\mathbf{I}; \theta)$  recursively by

$$\mathbf{I}^{(l)} = h(w_l \mathbf{I}^{(l-1)} + b_l), \quad (11.16)$$

for  $l = 1, \dots, L$ , where  $h(\cdot)$  is applied element-wise.  $\mathbf{I}^{(0)} = \mathbf{I}$ , and  $f(\mathbf{I}; \theta) = \mathbf{I}^{(L)}$ .  $\mathbf{I}^{(l)}$  consists of all the filtered images or feature maps at layer  $l$ , and the rows of  $w_l$  consist of all the filters as well as all the locations where the filters operate on  $\mathbf{I}^{(l-1)}$  to extract the features in  $\mathbf{I}^{(l)}$ . We assume that at the final layer  $L$ ,  $\mathbf{I}^{(L)}$  is reduced to a number (i.e., a  $1 \times 1$  feature map).  $w = (w_l, b_l, l = 1, \dots, L)$ . We can compare the compact equation (11.16) with the more detailed equation (11.5).

For piecewise linear  $h(\cdot)$ , such as  $h(r) = \max(0, r)$ , the function  $f(\mathbf{I}; \theta)$  is piecewise linear [160, 176]. Specifically,  $h(r) = \max(0, r) = 1(r > 0)r$ , where  $1(r > 0)$  is the indicator function that returns 1 if  $r > 0$  and 0 otherwise. Then

$$\mathbf{I}^{(l)} = s_l(\mathbf{I}; \theta)(w_l \mathbf{I}^{(l-1)} + b_l), \quad (11.17)$$

where

$$s_l(\mathbf{I}; \theta) = \text{diag}(\mathbb{1}(w_l \mathbf{I}^{(l-1)} + b_l > 0)), \quad (11.18)$$

i.e., a diagonal matrix of binary indicators (the indicator function is applied element-wise) [176]. Let  $s = (s_l, l = 1, \dots, L)$  consists of indicators at all the layers, then

$$f(\mathbf{I}; \theta) = \mathbf{B}_{s(\mathbf{I}; \theta)} \mathbf{I} + a_{s(\mathbf{I}; \theta)} \quad (11.19)$$

is piecewise linear, where

$$\mathbf{B}_s = \prod_{l=L}^1 s_l w_l, \quad (11.20)$$

and  $a_s$  can be similarly calculated.  $s(\mathbf{I}; \theta)$  partitions the image space of  $\mathbf{I}$  into exponentially many pieces [176] according to the value of  $s(\mathbf{I}; \theta)$ . The partition is recursive because  $s_l(\mathbf{I}; \theta)$  depends on  $s_{l-1}(\mathbf{I}; \theta)$ . The boundaries between the pieces are all linear. On each piece with  $s(\mathbf{I}; \theta) = s$ , where  $s$  on the right hand side denotes a particular value of  $s(\mathbf{I}; \theta)$ ,  $f(\mathbf{I}; \theta)$  is a linear function  $f(\mathbf{I}; \theta) = \mathbf{B}_s \mathbf{I} + a_s$ . The binary switches in  $s(\mathbf{I}; \theta)$  reconfigure the linear transformation according to (11.20).

$f(\mathbf{I}; \theta)$  generalizes three familiar structures in statistics:

(1) Generalized linear model (GLM). A GLM structure is a composition of a linear combination of the input variables and a non-linear link function. A ConvNet can be viewed as a recursion of this structure, where each component of  $\mathbf{I}^{(l)}$  is a GLM transformation of  $\mathbf{I}^{(l-1)}$ , with  $h$  being the link function.

(2) Linear spline. A one-dimensional linear spline is of the form  $y = \beta_0 + \sum_{k=1}^d \beta_k \max(0, x - a_k)$ , where  $a_k$  are the knots. The ConvNet  $f(\mathbf{I}; \theta)$  can be viewed as a multi-dimensional linear spline. The number of linear pieces is exponential in the number of layers [176]. Such a structure can approximate any continuous non-linear function by a large number of linear pieces.

(3) CART [20] and MARS [61]. In the classification and regression tree (CART) and the multivariate adaptive regression splines (MARS), the input domain is recursively partitioned. The linear pieces mentioned above are also recursively partitioned according to the values of  $s_l(\mathbf{I}; \theta)$  for  $l = 1, \dots, L$ . Moreover, MARS also makes use of the hinge function  $\max(0, r)$ .

For Gaussian reference  $q(\mathbf{I})$ , the energy function is

$$U(\mathbf{I}; \theta) = -f(\mathbf{I}; \theta) + \frac{1}{2\sigma^2} \|\mathbf{I}\|^2. \quad (11.21)$$

We can continue to use Langevin dynamics (12.76) to sample from  $p(\mathbf{I}; \theta)$ .

The parameter  $w$  can be learned by the stochastic gradient ascent algorithm [254]

$$w^{(t+1)} = w^{(t)} + \gamma_t \left[ \frac{1}{n} \sum_{i=1}^n \frac{\partial}{\partial w} f(\mathbf{I}_i; \theta) - \frac{1}{\tilde{n}} \sum_{i=1}^{\tilde{n}} \frac{\partial}{\partial w} f(\tilde{\mathbf{I}}_i; \theta) \right], \quad (11.22)$$

where again  $\gamma_t$  is the learning rate, and  $\{\tilde{\mathbf{I}}_i, i = 1, \dots, \tilde{n}\}$  are the synthetic images sampled from  $p(\mathbf{I}; w^{(t)})$ . This is again an analysis by synthesis scheme. This step shifts the probability density function  $p(\mathbf{I}; \theta)$ , or more specifically, the high probability regions or the low energy regions, from the synthetic images  $\{\tilde{\mathbf{I}}_i\}$  to the observed images  $\{\mathbf{I}_i\}$ .

In the sampling step, we need to compute  $\partial f(\mathbf{I}; \theta) / \partial \mathbf{I}$ . In the learning step, we need to compute  $\partial f(\mathbf{I}; \theta) / \partial w$ . Both derivatives can be calculated by the chain rule back-propagation, and they share the computations of  $\partial \mathbf{I}^{(l)} / \partial \mathbf{I}^{(l-1)}$ .

Our experiments show that the model is quite expressive. For example, we learn a 3-layer model. The first layer has 100  $15 \times 15$  filters with a sub-sampling size of 3 pixels. The second layer has 64  $5 \times 5$  filters with a sub-sampling size of 1. The third layer has 30  $3 \times 3$  filters with a sub-sampling size of 1. We learn a model (11.14) for each texture category from a single training image. Figure 11.11 displays some results. For each category, the first image is the training image, and the rest are 2 of the images generated by the learning algorithm. We use  $\tilde{n} = 16$  parallel chains for Langevin sampling. The number of Langevin iterations between every two consecutive updates of parameters is 10. The training images are of the size  $224 \times 224$ , whose intensities are within  $[0, 255]$ . We fix  $\sigma^2 = 1$  in the reference distribution  $q$ .

While the sparse FRAME model is interpretable in terms of the symbolic sketch of the images, the deep FRAME model is not interpretable with its multiple layers of dense connections in linear filtering. Perhaps the non-interpretability of the deep ConvNets is a fact we have to live with, very much like we find peace with quantum mechanics with its unitary linear evolution of the wave function and non-linear probabilistic collapsing of the wave function at measurement, as long as it is mathematically consistent and it gives correct predictions. The dense connections may be doing some implicit form of Bayesian model averaging without explicitly inferring latent variables whose uncertainties may be too large to be worthy of explicit inference, especially at the lower layers. On the other hand, at the higher layers, sparse connections and symbolic representations, as well as grammatical understanding [260] and logical reasoning, may naturally emerge, as the uncertainties become smaller. It would be interesting to find out how such sparse and symbolic representations arise from dense continuous vector representations.

### 11.1.7 Hopfield Auto-encoder

Consider the sparse FRAME model. Let us assume that the reference distribution  $q(\mathbf{I})$  is white noise with mean 0 and variance  $\sigma^2 = 1$ . The energy function is



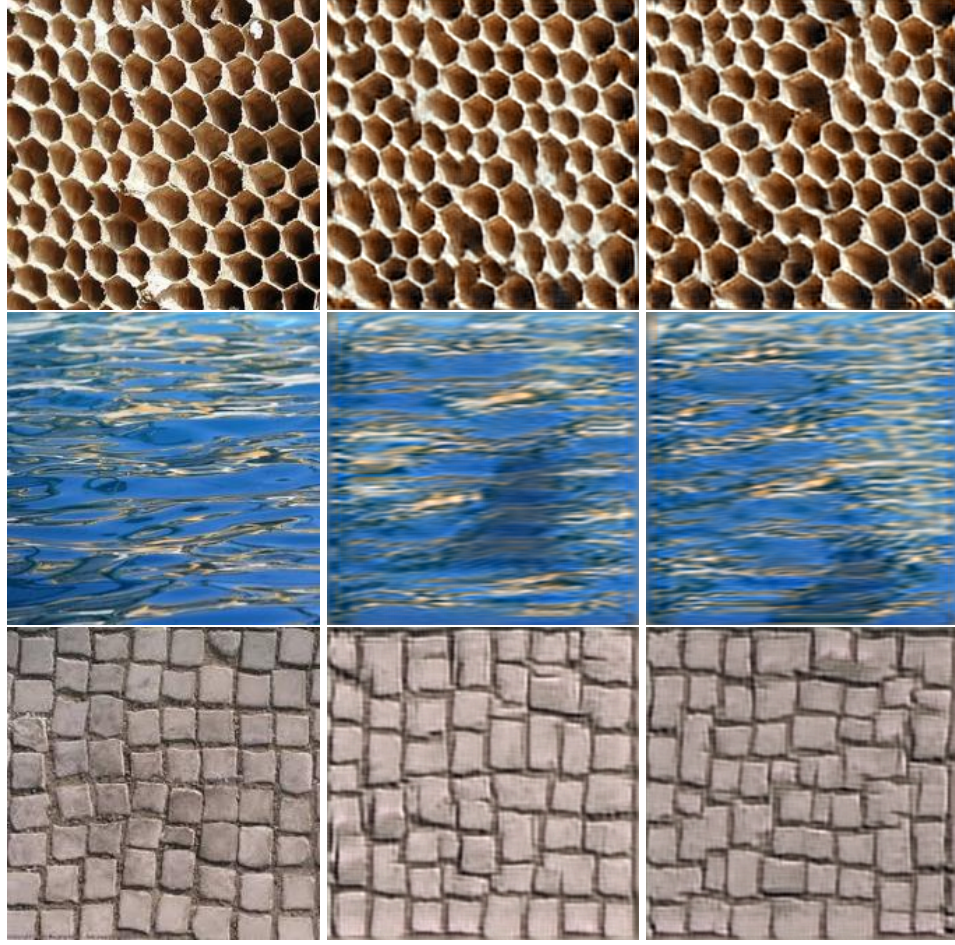


Figure 11.11: Reprinted with permission from [244]. Generating texture patterns. For each category, the first image ( $224 \times 224$ ) is the training image, and the rest are 2 of the images generated by the learning algorithm.

$$U(\mathbf{I}) = \frac{1}{2} \|\mathbf{I}\|^2 - \sum_{j=1}^m w_j h(\langle \mathbf{I}, B_{k_j, x_j} \rangle). \quad (11.23)$$

This energy function can be multi-modal, and each local minimum  $\hat{\mathbf{I}}$  satisfies  $U'(\hat{\mathbf{I}}) = 0$ . Thus,

$$\hat{\mathbf{I}} = \sum_{j=1}^m w_j h'(\langle \hat{\mathbf{I}}, B_{k_j, x_j} \rangle) B_{k_j, x_j}. \quad (11.24)$$

This reveals an auto-encoder [14, 220] hidden in the local modes:

$$\text{Encoding : } c_j = w_j h'(\langle \hat{\mathbf{I}}, B_{k_j, x_j} \rangle), \quad (11.25)$$

$$\text{Decoding : } \hat{\mathbf{I}} = \sum_{i=1}^n c_j B_{k_j, x_j}, \quad (11.26)$$

where (11.25) encodes  $\hat{\mathbf{I}}$  by  $(c_j)$ , and (11.26) reconstructs  $\hat{\mathbf{I}}$  from  $(c_j)$ .  $B_{k_j, x_j}$  serves as both bottom-up filter in (11.25) and top-down basis function in (11.26). We call this auto-encoder the Hopfield auto-encoder

because  $\hat{\mathbf{I}}$  is a local minimum of the energy function (11.23). Hopfield [97] proposes that the local energy minima may be used for content-addressable memory.

The Hopfield auto-encoder also presents itself in the deep convolutional energy-based model (11.14) [244]. The energy function of the model is  $\|\mathbf{I}\|^2/2 - f(\mathbf{I}; \theta)$ . The local minima satisfies the Hopfield auto-encoder  $\hat{\mathbf{I}} = f'(\hat{\mathbf{I}}; \theta)$ , or more specifically,

$$\text{Encoding : } s = s(\hat{\mathbf{I}}; \theta), \quad (11.27)$$

$$\text{Decoding : } \hat{\mathbf{I}} = \mathbf{B}_s. \quad (11.28)$$

The encoding process is a bottom-up computation of the indicators at different layers  $s_l = s_l(\mathbf{I}; \theta)$ , for  $l = 1, \dots, L$ , where  $w_l$  plays the role of filters, see equation (11.18). The decoding process is a top-down computation for reconstruction, where  $s_l$  plays the role of coefficients, and  $w_l$  plays the role of basis functions. See equation (11.20). The encoding process detects the patterns corresponding to the filters, and then the decoding process reconstructs the image using the detected filters as the basis functions.

Now let us think about a simple example. Suppose we fit an energy-based neural network model which has 2 fully connected hidden layers and we use Gaussian as reference distribution, i.e. the energy function can be written as the following form:

$$U(\mathbf{I}) = \frac{1}{2}\|\mathbf{I}\|^2 - (W_3 h(W_2 h(W_1 \mathbf{I} + b_1) + b_2) + b_3). \quad (11.29)$$

In this equation, we treat our  $n$ -dimensional input  $\mathbf{I}$  as a  $n \times 1$  matrix. Suppose we have  $n_1$  neurons in the first layer and  $n_2$  neurons in the second layer, then  $W_1$  is a  $n_1 \times n$  matrix,  $W_2$  is a  $n_2 \times n_1$  matrix and  $W_3$  is a  $1 \times n_2$  matrix (the final output shape 1 corresponds to the single channel energy output).  $b_1, b_2, b_3$  has shape  $n_1 \times 1, n_2 \times 1, 1 \times 1$ . We use  $h(\cdot)$  to denote the activation function, which can be Sigmoid, ReLU, Softplus, etc. Then at each local minimum, we have  $\frac{\partial U}{\partial \mathbf{I}} = 0$ , which gives us the following equation:

$$\begin{aligned} \mathbf{I}^\top &= W_3 S_2 W_2 S_1 W_1 \\ S_1 &= \text{diag}\left(\frac{\partial h(W_1 \mathbf{I} + b_1)}{\partial (W_1 \mathbf{I} + b_1)}\right) \\ S_2 &= \text{diag}\left(\frac{\partial h(W_2 h(W_1 \mathbf{I} + b_1) + b_2)}{\partial (W_2 h(W_1 \mathbf{I} + b_1) + b_2)}\right) \end{aligned} \quad (11.30)$$

$S_i$  is our encoding in this case, which is a diagonal matrix whose diagonal elements are the gradients of the activation function we use. Then we can define our bases and rewrite 11.30 as:

$$B_1 = S_1 W_1 \quad \mathbf{I}^\top = W_3 S_2 W_2 B_1 \quad (11.31)$$

$$B_2 = S_2 W_2 S_1 W_1 \quad \mathbf{I}^\top = W_3 B_2 \quad (11.32)$$

In other words, we can write  $\mathbf{I}$  as the reconstruction of the bases function at each layer. Note that the bases  $B_i$  we use here are not fixed, but is a function which depends on the input itself, i.e. the model first encode the data into  $S_i$ 's, then these  $S_i$ 's will generate  $B_i$ 's and at local minimum points, these  $B_i$ 's reconstruct the input data.

We show the result from a toy example in Figure 11.12. Here we assume the true data lies in a circle. Our neural network has 10 neurons at the first layer and 8 neurons at the second layer. As for the activation function, we use Softplus. Shown in 11.33, Softplus can be seen as a smooth version at ReLU, which is

differentiable everywhere (ReLU is not differentiable at 0). We choose it because our analysis relies on the gradients of the energy function (11.30 does not apply to non-differentiable points). In Figure 11.12 (c) and (d), we visualize the basis got from different layers. Remember as we discussed above, currently, a "basis" we talked about here is actually a function that maps a two-dimensional input to a two-dimensional output. Therefore, given an input data point, for each basis function, we can calculate a two-dimensional output point. In Figure 11.12 (c) and (d), we calculate all the basis functions on all the training data points. We then plot all these output points on a two-dimensional space and use the same color to represent points from the same basis function. One can understand this as a visualization of given that the original data points locate on a circle, how the output points from a certain basis lie in the space. From Figure 11.12 (c), we can see that the bases of the first layer are straight line segments. This is because the  $S_1$  is a diagonal matrix, thus it can only re-scale the fixed weight  $W_1$  instead of mixing them together. But the bases of the second layer can actually have different shapes. As shown in (d), we can see in this case, the model learns some basis with circle-like shape. This is closely related to the space partition (We can see that the model finds some circle-like partition lines in (b)). In (e), we can see that the reconstructed values agree with the input data at the local minimum points.

$$\begin{aligned} h(a) &= \ln(1 + \exp(a)) \\ h'(a) &= \frac{1}{1 + \exp(-a)} \end{aligned} \quad (11.33)$$

One thing we want to note here is that although we find some interesting shapes for high layer bases in this toy example. In real applications. It is very unlikely that we can get bases that carry any explicitly interpretable meanings. Actually, 11.12 (d) shows that although our data only lie on a circle that has a radius of 0.6, the bases can take value in much wider ranges (-20 to 20). That is because we only require the weighted sum of these bases to reconstruct the original data. And they only reconstruct the data which lies at the local minimum of the fitted energy function. During the training process, our only goal is to match the statistics at the final layer of our neural network instead of reconstructing each data point. Also, under different initialization, we may get totally different space partition and thus different bases function. As shown in Figure 11.13, under another initialization, we actually have some triangle-like bases.

The relationship between auto-encoders and energy-based models [128] has been investigated by [219] and [207] for the restricted Boltzmann machine and its extensions [93]. A regularized auto-encoder is a special form of score matching estimator [103]. The Hopfield auto-encoder was first elucidated by [244].

### 11.1.8 Multigrid Sampling and Modeling

In the high-dimensional space, e.g. image space, the model can be highly multi-modal. The MCMC in general and the Langevin dynamics, in particular, may have difficulty traversing different modes and it may be very time-consuming to converge. A simple and popular modification of the maximum likelihood learning is the contrastive divergence (CD) learning [91], where we obtain the synthesized example by initializing a finite-step MCMC from the observed example. The CD learning is related to score matching estimator [103, 104] and auto-encoder [5, 207, 219]. Such a method has the ability to handle large training datasets via mini-batch training. However, bias may be introduced in the learned model parameters in that the synthesized images can be far from the fair examples of the current model. A further modification of CD is persistent CD [211], where at the initial learning epoch the MCMC is still initialized from the observed examples, while in each subsequent learning epoch, the finite-step MCMC is initialized from the

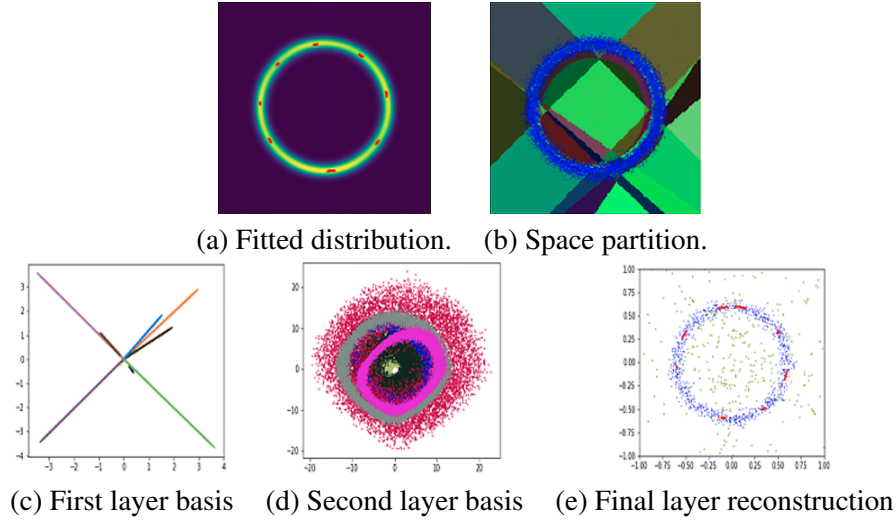


Figure 11.12: Results for Hopfield auto-encoder. (a) The fitted energy function. Red points are the fitted local minimum. (b) Space partition. (c)(d) Plotting the bases at each layer. We calculate the basis values at each training data. Each point in the figure is the value of a certain basis calculated from one training data. Different color indicates different basis. (e) Plotting the reconstruction at the final layer. The red points are reconstructed from local minimum points; the green points are reconstructed from non-minimum points; the blue points are true values. We can see that at the local minimum, reconstruction and true values agree with each other.

synthesized example of the previous epoch. The resulting synthesized examples can be less biased by the observed examples. However, the persistent chains may still have difficulty traversing different modes of the learned model.

A multi-grid sampling and learning method is developed in [65] to address the above challenges under the constraint of finite budget MCMC. Specifically, each training image is repeatedly down-scaled to get its multi-grid versions. A separate energy-based model is learned at each grid. Within each iteration of the learning algorithm, for each observed training image, the corresponding synthesized images are generated at multiple grids. We can initialize the finite-step MCMC sampling from the minimal  $1 \times 1$  version of the training image, and the synthesized image at each grid serves to initialize the finite-step MCMC that samples from the model of the subsequent finer grid. See Figure 11.14 for an illustration, where the images are sampled sequentially at 3 grids, with 30 steps of Langevin dynamics at each grid. After obtaining the synthesized images at the multiple grids, the models at the multiple grids are updated separately and simultaneously based on the differences between the synthesized images and the observed training images at different grids.

Unlike the original CD or persistent CD, the learned models are capable of generating new synthesized images from scratch with a fixed budget MCMC, because we only need to initialize the MCMC by sampling from the one-dimensional histogram of the  $1 \times 1$  versions of the training images. See Figures 11.15 and 11.16 for generated samples learned from large image datasets.

The learned energy-based model is a bottom-up ConvNet that consists of multiple layers of features. These features can be used for subsequent tasks such as classification. The learned models can also be used as a prior distribution for inpainting, as illustrated by Figure 11.17. See [65] for experiment details and numerical evaluations.



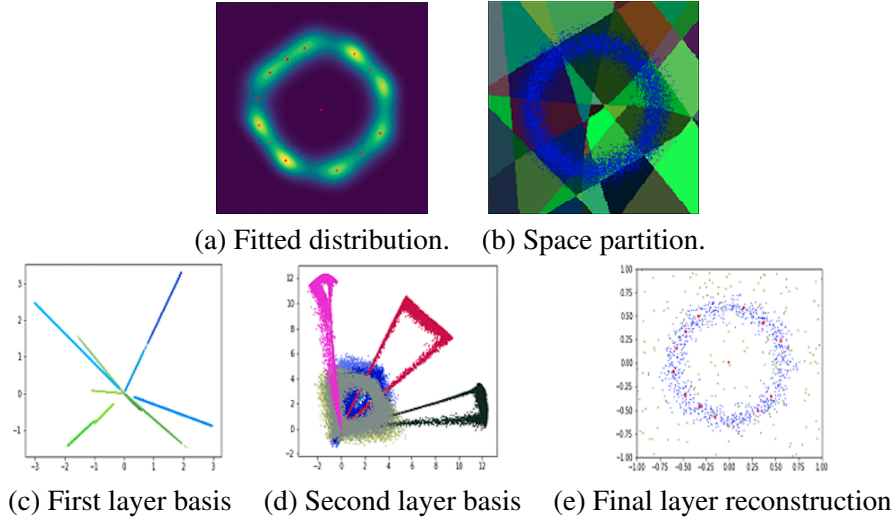


Figure 11.13: Results for Hopfield auto-encoder from another initialization. (a) The fitted energy function. Red points are the fitted local minimum. (b) Space partition. (c)(d) Plotting the bases at each layer. We calculate the basis values at each training data. Each point in the figure is the value of a certain basis calculated from one training data. Different color indicates different basis. (e) Plotting the reconstruction at the final layer. The red points are reconstructed from local minimum points; the green points are reconstructed from non-minimum points; the blue points are true values. We can see that at the local minimum, reconstruction and true values agree with each other.

### 11.1.9 Diffusion Recovery Likelihood

Although the multigrid sampling and modeling method enable us to generate new samples from energy-based models (EBMs) learned on high dimensional datasets such as images, two challenges remain for training EBMs on high-dimensional datasets. First, learning EBMs by maximum likelihood requires Markov Chain Monte Carlo (MCMC) to generate samples from the model, which can be extremely expensive. Second, the energy potentials learned with non-convergent MCMC do not have a valid steady-state, in the sense that samples from long-run Markov chains can differ greatly from observed samples, making it difficult to evaluate the learned energy potentials.

[67] proposes a *diffusion recovery likelihood* method to tackle the challenge of training EBMs directly on a dataset by instead learning a sequence of EBMs for the marginal distributions of the diffusion process. Specifically, assume a sequence of perturbed observations  $\mathbf{x}_0, \mathbf{x}_1, \dots, \mathbf{x}_T$  such that

$$\mathbf{x}_0 \sim p_{\text{data}}(\mathbf{x}); \quad \mathbf{x}_{t+1} = \sqrt{1 - \sigma_{t+1}^2} \mathbf{x}_t + \sigma_{t+1} \epsilon_{t+1}, \quad t = 0, 1, \dots, T-1. \quad (11.34)$$

The scaling factor  $\sqrt{1 - \sigma_{t+1}^2}$  ensures that the sequence is a spherical interpolation between the observed sample and Gaussian white noise. Let  $\mathbf{y}_t = \sqrt{1 - \sigma_{t+1}^2} \mathbf{x}_t$ , and we assume a sequence of marginal EBMs on the perturbed data

$$p_{\theta}(\mathbf{y}_t) = \frac{1}{Z_{\theta,t}} \exp(f_{\theta}(\mathbf{y}_t, t)), \quad (11.35)$$

where  $f_{\theta}(\mathbf{y}_t, t)$  is defined by a neural network conditioned on  $t$ . The sequence of marginal EBMs can be

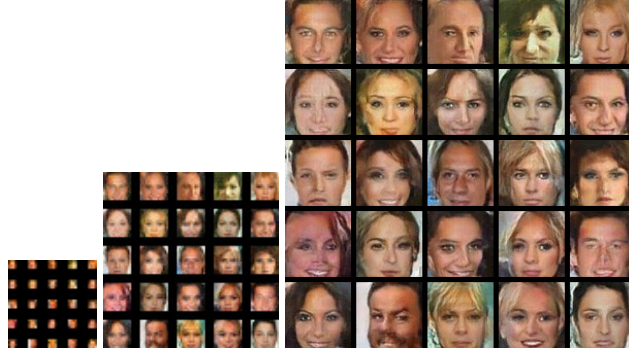


Figure 11.14: Reprinted with permission from [65]. Synthesized images at multi-grids. From left to right:  $4 \times 4$  grid,  $16 \times 16$  grid and  $64 \times 64$  grid. Synthesized image at each grid is obtained by 30 step Langevin sampling initialized from the synthesized image at the previous coarser grid, beginning with the  $1 \times 1$  grid.

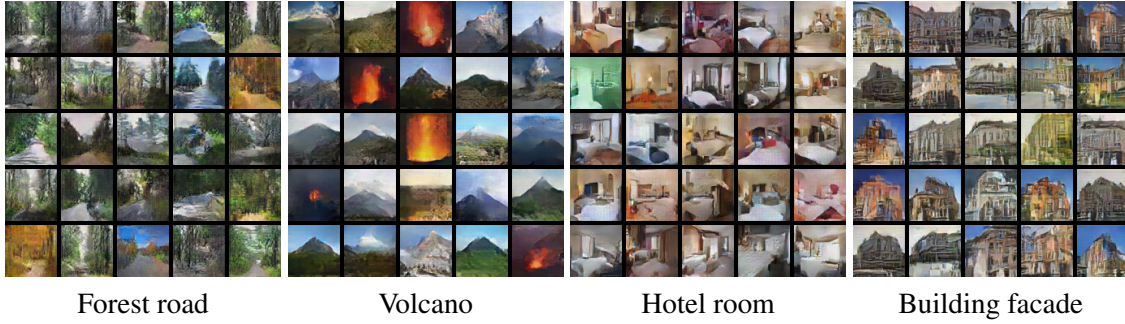


Figure 11.15: Reprinted with permission from [65]. Synthesized images from models learned by multi-grid method from 4 categories of MIT places205 datasets.

learned with recovery likelihoods that are defined as the conditional distributions that invert the diffusion process, which can be derived by equations 11.34 and 11.35:

$$p_{\theta}(\mathbf{y}_t | \mathbf{x}_{t+1}) = \frac{1}{\tilde{Z}_{\theta,t}(\mathbf{x}_{t+1})} \exp \left( f_{\theta}(\mathbf{y}_t, t) - \frac{1}{2\sigma_{t+1}^2} \|\mathbf{x}_{t+1} - \mathbf{y}_t\|^2 \right), \quad t = 0, 1, \dots, T-1. \quad (11.36)$$

Compared to the standard maximum likelihood estimation (MLE) of EBMs, learning marginal EBMs by diffusion recovery likelihood only requires sampling from the conditional distributions in equation 11.36, which is much easier than sampling from the marginal distributions due to the additional quadratic terms. After learning the marginal EBMs, we can generate synthesized images by a sequence of conditional samples initialized from the Gaussian white noise distribution using MCMC techniques such as Langevin sampling:

$$\mathbf{y}_t^{\tau+1} = \mathbf{y}_t^{\tau} + \frac{b^2 \sigma_t^2}{2} (\nabla_{\mathbf{y}} f_{\theta}(\mathbf{y}_t^{\tau}, t) + \frac{1}{\sigma_t^2} (\mathbf{x}_{t+1} - \mathbf{y}_t^{\tau})) + b \sigma_t \epsilon^{\tau}. \quad (11.37)$$

Figure 11.18 shows an illustration on a 2D toy example. The framework of recovery likelihood was originally proposed in [15]. [67] adapt it to learning the sequence of marginal EBMs from the diffusion data. Figure 11.19 display uncured samples generated from learned models on large image datasets.

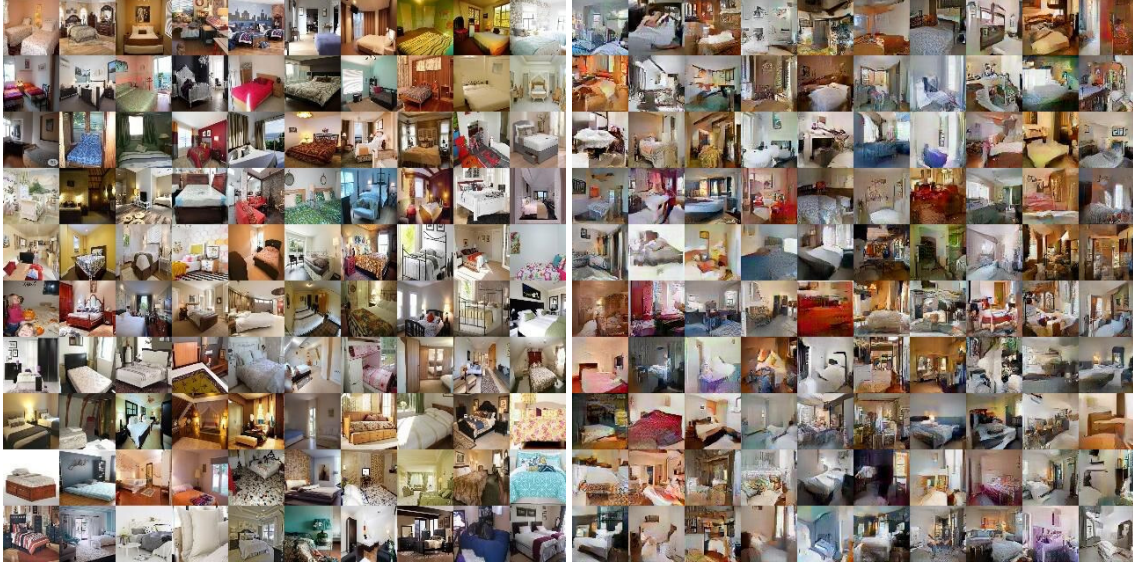


Figure 11.16: Reprinted with permission from [65]. Learning the multi-grid models from the LSUN bedroom dataset. Left: random samples of training examples. Right: synthesized examples generated by the learned models.

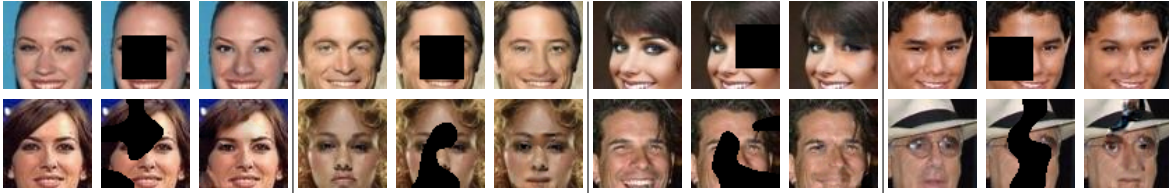


Figure 11.17: Reprinted with permission from [65]. Inpainting examples on CelebA dataset. In each block from left to right: the original image; masked input; inpainted image by multi-grid method.

### 11.1.10 Adversarial Interpretation

The deep convolutional energy-based model (11.14) can be written as

$$p(\mathbf{I}; \theta) = \frac{1}{Z(\theta)} \exp[-U(\mathbf{I}; \theta)], \quad (11.38)$$

where the energy function  $U(\mathbf{I}; \theta) = -f(\mathbf{I}; \theta) + \frac{1}{2\sigma^2} \|\mathbf{I}\|^2$ . The update of  $w$  is based on  $L'(\theta)$  which can be approximated by

$$\frac{1}{\tilde{n}} \sum_{i=1}^{\tilde{n}} \frac{\partial}{\partial w} U(\tilde{\mathbf{I}}_i; \theta) - \frac{1}{n} \sum_{i=1}^n \frac{\partial}{\partial w} U(\mathbf{I}_i; \theta), \quad (11.39)$$

where  $\{\tilde{\mathbf{I}}_i, i = 1, \dots, \tilde{n}\}$  are the synthetic images that are generated by the Langevin dynamics. At the zero temperature limit, the Langevin dynamics becomes gradient descent:

$$\tilde{\mathbf{I}}_{\tau+1} = \tilde{\mathbf{I}}_{\tau} - \delta \frac{\partial}{\partial \tilde{\mathbf{I}}} U(\tilde{\mathbf{I}}_{\tau}; \theta). \quad (11.40)$$

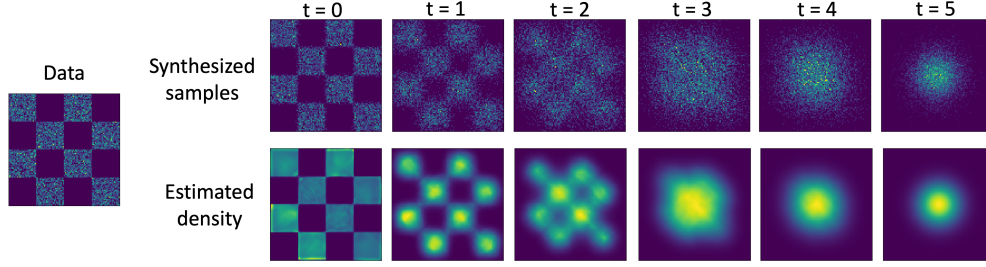


Figure 11.18: Reprinted with permission from [67]. Illustration of diffusion recovery likelihood on 2D checkerboard example. *Top*: progressively generated samples. *Bottom*: estimated marginal densities.

Consider the value function

$$V(\tilde{\mathbf{I}}_i, i = 1, \dots, \tilde{n}; \theta) = \frac{1}{\tilde{n}} \sum_{i=1}^{\tilde{n}} U(\tilde{\mathbf{I}}_i; \theta) - \frac{1}{n} \sum_{i=1}^n U(\mathbf{I}_i; \theta). \quad (11.41)$$

The updating of  $w$  is to increase  $V$  by shifting the low energy regions from the synthetic images  $\{\tilde{\mathbf{I}}_i\}$  to the observed images  $\{\mathbf{I}_i\}$ , whereas the updating of  $\{\tilde{\mathbf{I}}_i, i = 1, \dots, \tilde{n}\}$  is to decrease  $V$  by moving the synthetic images towards the low energy regions. This is an adversarial interpretation of the learning and sampling algorithm. It can also be considered a generalization of the herding method [227] from exponential family models to general energy-based models.

If we recruit a generator model  $g(\mathbf{I}; \tilde{w})$  as a sampler, then the energy-based model and the generator model play a minimax game with the value function

$$V(\tilde{w}; \theta) = \mathcal{E}_{\tilde{w}}[U(\tilde{\mathbf{I}}; \theta)] - \mathcal{E}_{\text{data}}[U(\mathbf{I}; \theta)], \quad (11.42)$$

where  $\mathcal{E}_{\tilde{w}}$  is the expectation with respect to the generator model with parameter  $\tilde{w}$ , and  $\mathcal{E}_{\text{data}}[U(\mathbf{I}; \theta)] = \frac{1}{n} \sum_{i=1}^n U(\mathbf{I}_i; \theta)$ . This is related to [7].

## 11.2 Generator Network

This section studies the fundamental problem of learning and inference in the generator network [74]. The generator network is based on a top-down ConvNet. We recognize that the generator network is a non-linear generalization of the factor analysis model and develop the alternating back-propagation algorithm as the non-linear generalization of the alternating regression scheme of the Rubin-Thayer EM algorithm for fitting the factor analysis model. The alternating back-propagation algorithm iterates the inferential back-propagation for inferring the latent factors and the learning back-propagation for updating the parameters. Both back-propagation steps share most of their computing steps in the chain rule calculations.

The learning algorithm is perhaps the most canonical algorithm for training the generator network. It is based on maximum likelihood, which is theoretically the most accurate estimator. The maximum likelihood learning seeks to explain and charge the whole dataset uniformly, so that there is little concern of under-fitting or biased fitting.

As an unsupervised learning algorithm, the alternating back-propagation algorithm is a natural generalization of the original back-propagation algorithm for supervised learning. It adds an inferential back-propagation step to the learning back-propagation step, with minimal overhead in coding and affordable



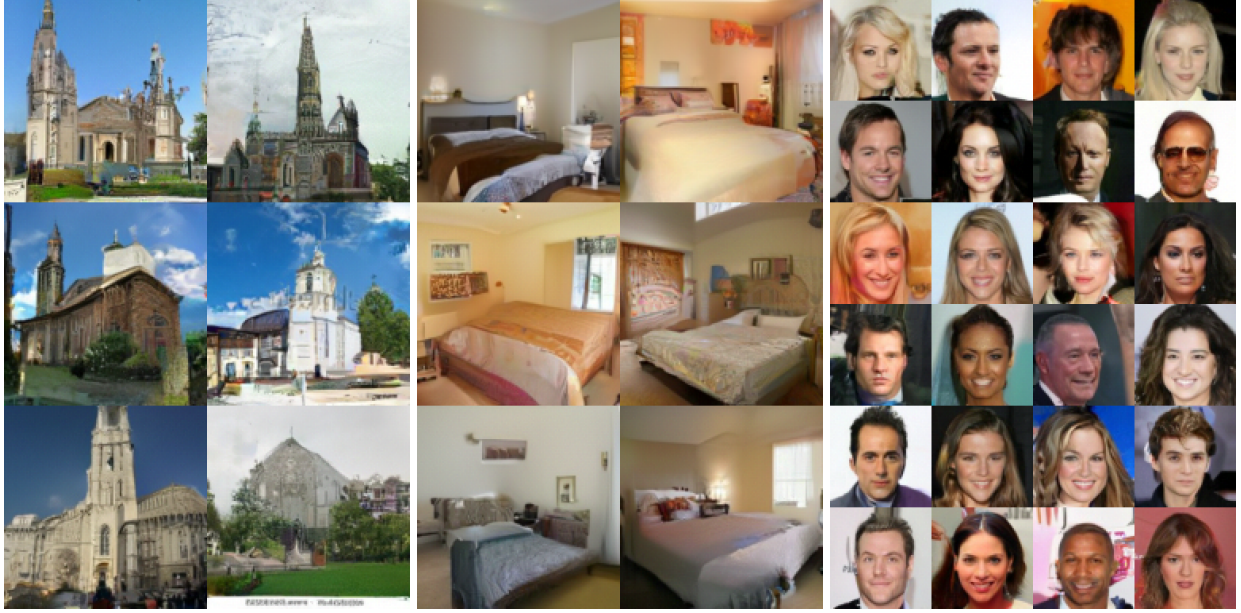


Figure 11.19: Reprinted with permission from [67]. Generated samples on LSUN 128<sup>2</sup> church\_outdoor (left), LSUN 128<sup>2</sup> bedroom (center) and CelebA 64<sup>2</sup> (right).

overhead in computing. The inferential back-propagation seeks to perform accurate explaining-away inference of the latent factors.

### 11.2.1 Factor Analysis

Let  $\mathbf{I}$  be a  $D$ -dimensional observed data vector, such as an image. Let  $Z$  be the  $d$ -dimensional vector of continuous latent factors,  $Z = (z_k, k = 1, \dots, d)$ . The traditional factor analysis model is  $\mathbf{I} = WZ + \epsilon$ , where  $W$  is  $D \times d$  matrix, and  $\epsilon$  is a  $D$ -dimensional error vector or the observational noise. We assume that  $Z \sim N(0, I_d)$ , where  $I_d$  stands for the  $d$ -dimensional identity matrix. We also assume that  $\epsilon \sim N(0, \sigma^2 I_D)$ , i.e., the observational errors are Gaussian white noises. There are three perspectives to view  $W$ .

(1) *Basis vectors.* Write  $W = (W_1, \dots, W_d)$ , where each  $W_k$  is a  $D$ -dimensional column vector. Then  $\mathbf{I} = \sum_{k=1}^d z_k W_k + \epsilon$ , i.e.,  $W_k$  are the basis vectors and  $z_k$  are the coefficients.

(2) *Loading matrix.* Write  $W = (w_1, \dots, w_D)^\top$ , where  $w_j^\top$  is the  $j$ -th row of  $W$ . Then  $\mathbf{I}_j = \langle w_j, Z \rangle + \epsilon_j$ , where  $\mathbf{I}_j$  and  $\epsilon_j$  are the  $j$ -th components of  $\mathbf{I}$  and  $\epsilon$  respectively. Each  $\mathbf{I}_j$  is a loading of the  $d$  factors where  $w_j$  is a vector of loading weights, indicating which factors are important for determining  $\mathbf{I}_j$ .  $W$  is called the loading matrix.

(3) *Matrix factorization.* Suppose we observe  $\mathbf{I} = (\mathbf{I}_1, \dots, \mathbf{I}_n)$ , whose factors are  $\mathbf{Z} = (Z_1, \dots, Z_n)$ , then  $\mathbf{I} \approx W\mathbf{Z}$ .

The factor analysis model can be learned by the Rubin-Thayer EM algorithm, which involves alternating regressions of  $Z$  on  $\mathbf{I}$  in the E-step and of  $\mathbf{I}$  on  $Z$  in the M-step [139, 191].

The factor analysis model is the prototype of many subsequent models that generalize the prior model of  $z$ .

(1) *Independent component analysis* [105],  $d = D$ ,  $\epsilon = 0$ , and  $z_k$  are assumed to follow independent heavy tailed distributions.

(2) *Sparse coding* [174],  $d > D$ , and  $Z$  is assumed to be a redundant but sparse vector, i.e., only a small number of  $z_k$  are non-zero or significantly different from zero.

(3) *Non-negative matrix factorization* [132], it is assumed that  $z_k \geq 0$ .

### 11.2.2 Non-linear Factor Analysis

The generator network is a non-linear generalization of factor analysis. Factor analysis is a prototype model in unsupervised learning of distributed representations. There are two directions one can pursue in order to generalize the factor analysis model. One direction is to generalize the prior model or the prior assumption about the latent factors. This led to methods such as independent component analysis [105], sparse coding [174], non-negative matrix factorization [132], matrix factorization and completion for recommender systems [122], etc.

The other direction to generalize the factor analysis model is to generalize the mapping from the continuous latent factors to the observed signal. The generator network is an example in this direction. It generalizes the linear mapping in factor analysis to a non-linear mapping that is defined by a convolutional neural network (ConvNet or CNN) [44, 124, 127]. It has been shown that the generator network is capable of generating realistic images [38, 182].

The generator network is a fundamental representation of knowledge, and it has the following properties:

(1) *Analysis*: The model disentangles the variations in the observed signals into independent variations of latent factors.

(2) *Synthesis*: The model can synthesize new signals by sampling the factors from the known prior distribution and transforming the factors into the signal.

(3) *Embedding*: The model embeds the high-dimensional non-Euclidean manifold formed by the observed signals into the low-dimensional Euclidean space of the latent factors, so that linear interpolation in the low-dimensional factor space results in non-linear interpolation in the data space.

In addition to generalizing the prior model of the latent factors  $Z$ , we can also generalize the mapping from  $Z$  to  $\mathbf{I}$ . Here, we consider the generator network model [74] that retains the assumptions that  $d < D$ ,  $Z \sim \mathcal{N}(0, I_d)$ , and  $\epsilon \sim \mathcal{N}(0, \sigma^2 I_D)$  as in traditional factor analysis, but generalizes the linear mapping  $WZ$  to a non-linear mapping  $g(Z; \theta)$ , where  $f$  is a ConvNet, and  $W$  collects all the connection weights and bias terms of the ConvNet. Then the model becomes

$$\begin{aligned} \mathbf{I} &= g(Z; \theta) + \epsilon, \\ Z &\sim \mathcal{N}(0, I_d), \epsilon \sim \mathcal{N}(0, \sigma^2 I_D), d < D. \end{aligned} \quad (11.43)$$

The reconstruction error is  $\|\mathbf{I} - g(Z; \theta)\|^2$ .

Although  $g(Z; \theta)$  can be any non-linear mapping, the ConvNet parameterization of  $g(Z; \theta)$  makes it particularly close to the original factor analysis. Specifically, we can write the top-down ConvNet as follows:

$$Z^{(l-1)} = g_l(W_l Z^{(l)} + b_l), \quad (11.44)$$

where  $g_l$  is element-wise non-linearity at layer  $l$ ,  $W_l$  is the matrix of connection weights,  $b_l$  is the vector of bias terms at layer  $l$ , and  $\theta = (W_l, b_l, l = 1, \dots, L)$ .  $Z^{(0)} = g(Z; \theta)$ , and  $Z^{(L)} = \mathbf{I}$ . The top-down ConvNet (11.44) can be considered a recursion of the  $g(Z; \theta)$  original factor analysis model, where the factors at the layer  $l - 1$  are obtained by the linear superposition of the basis vectors or basis functions that are column vectors of  $W_l$ , with the factors at the layer  $l$  serving as the coefficients of the linear superposition. In the case of ConvNet, the basis functions are shift-invariant versions of one another, like wavelets.

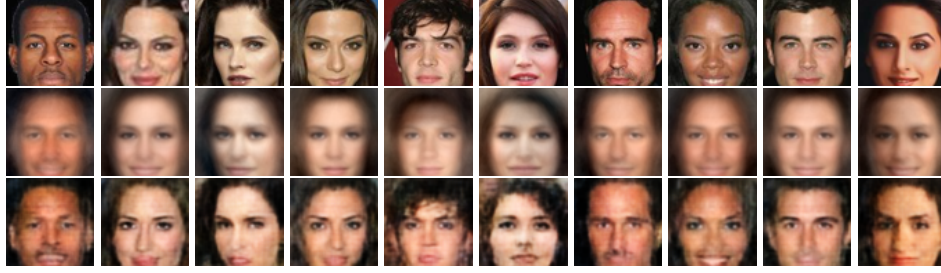


Figure 11.20: Reprinted with permission from [85]. Comparison between generator as non-linear and PCA as linear factor analysis. *Row 1*: original testing images. *Row 2*: reconstructions by PCA eigenvectors learned from training images. *Row 3*: reconstructions by the generator learned from training images.

### 11.2.3 Learning by Alternating Back-Propagation

The factor analysis model can be learned by the Rubin-Thayer EM algorithm [36, 191], where both the E-step and the M-step are based on multivariate linear regression. Inspired by this algorithm, we propose an alternating back-propagation algorithm for learning the generator network that iterates the following two-steps:

- (1) *Inferential back-propagation*: For each training example, infer the continuous latent factors by Langevin dynamics or gradient descent.
- (2) *Learning back-propagation*: Update the parameters given the inferred latent factors by gradient descent.

The Langevin dynamics [164] is a stochastic sampling counterpart of gradient descent. The gradient computations in both steps are powered by back-propagation. Because of the ConvNet structure, the gradient computation in step (1) is actually a by-product of the gradient computation in step (2).

Given the factors, the learning of the ConvNet is a supervised learning problem [44] that can be accomplished by the learning back-propagation. With factors unknown, the learning becomes an unsupervised problem, which can be solved by adding the inferential back-propagation to the learning process. We shall show that the alternating back-propagation algorithm can learn realistic generator models of natural images.

The alternating back-propagation algorithm follows the tradition of alternating operations in unsupervised learning, such as alternating linear regression in the EM algorithm for factor analysis, alternating least squares algorithm for matrix factorization [117, 122], and alternating gradient descent algorithm for sparse coding [174]. All these unsupervised learning algorithms alternate an inference step and a learning step, as is the case with alternating back-propagation.

The inferential back-propagation solves an inverse problem by an explaining-away process, where the latent factors compete with each other to explain each training example.

Figure 11.20 depicts the reconstructions of face images by linear PCA and non-linear generator net. For PCA, we learn the  $d$  eigenvectors from the training images, and then project the testing images on the learned eigenvectors for reconstruction. The generator is learned by alternating back-propagation. We infer the  $d$  dimensional latent factors  $Z$  using inferential back-propagation, and then reconstruct the testing image by  $g(Z; \theta)$  using the inferred  $Z$  and the learned  $W$ .

The generator network is a directed graphical model, where  $\mathbf{I}$  can be readily generated by first sampling  $Z$  from its known prior distribution  $N(0, I_d\theta)$  and then transforming  $Z$  to  $\mathbf{I}$  via  $g$ . The joint density is  $P(Z, \mathbf{I}; \theta) = P(Z)P(\mathbf{I}|Z; \theta)$ , and

$$\log P(Z, \mathbf{I}; \theta) = -\frac{1}{2\sigma^2}\|\mathbf{I} - g(Z; \theta)\|^2 - \frac{1}{2}\|Z\|^2 + \text{constant},$$

where the constant term is independent of  $Z$ ,  $\mathbf{I}$  and  $W$ .

The marginal density is obtained by integrating out the latent factors  $Z$ , i.e.,  $P(\mathbf{I}; \theta) = \int P(Z, \mathbf{I}; \theta) dZ$ . The inference of  $Z$  given  $\mathbf{I}$  is based on the posterior density  $P(Z|\mathbf{I}; \theta) = P(Z, \mathbf{I}; \theta)/P(\mathbf{I}; \theta) \propto P(Z, \mathbf{I}; \theta)$  as a function of  $Z$ .

For the training data  $\{\mathbf{I}_i, i = 1, \dots, n\}$ , the generator net can be trained by maximizing the log-likelihood

$$L(\theta) = \frac{1}{n} \sum_{i=1}^n \log P(\mathbf{I}_i; \theta). \quad (11.45)$$

For a large sample,  $W$  is obtained by minimizing the Kullback-Leibler divergence  $\text{KL}(P_{\text{data}}|P)$  from the data distribution  $P_{\text{data}}$  to the model distribution  $P$ .

The gradient of  $L(\theta)$  is obtained according to the following identity

$$\begin{aligned} \frac{\partial}{\partial W} \log P(\mathbf{I}; \theta) &= \frac{1}{P(\mathbf{I}; \theta)} \frac{\partial}{\partial W} \int P(\mathbf{I}, Z; \theta) dZ \\ &= \frac{1}{P(\mathbf{I}; \theta)} \int \left[ \frac{\partial}{\partial W} \log P(\mathbf{I}, Z; \theta) \right] P(\mathbf{I}, Z; \theta) dZ \\ &= \int \left[ \frac{\partial}{\partial W} \log P(\mathbf{I}, Z; \theta) \right] \frac{P(\mathbf{I}, Z; \theta)}{P(\mathbf{I}; \theta)} dZ \\ &= \mathcal{E}_{P(Z|\mathbf{I}; \theta)} \left[ \frac{\partial}{\partial W} \log P(Z, \mathbf{I}; \theta) \right]. \end{aligned} \quad (11.46)$$

The above identity underlies the EM algorithm, where  $\mathcal{E}_{P(Z|\mathbf{I}; \theta)}$  is the expectation with respect to the posterior distribution of the latent factors  $P(Z|\mathbf{I}; \theta)$ , and is computed in the E-step. The usefulness of identity (11.46) lies in the fact that the derivative of the complete-data log-likelihood  $\log P(Z, \mathbf{I}; \theta)$  on the right-hand side can be obtained in closed form. In the EM algorithm, the M-step maximizes the expectation of  $\log P(Z, \mathbf{I}; \theta)$  with respect to the current posterior distribution of the latent factors.

In general, the expectation in (11.46) is analytically intractable and has to be approximated by MCMC that samples from the posterior  $P_G(Z|\mathbf{I}; \theta)$ , such as the Langevin inference dynamics, which iterates

$$Z_{\tau+1} = Z_{\tau} + \frac{\delta^2}{2} \frac{\partial}{\partial Z} \log P(Z_{\tau}, \mathbf{I}; \theta) + \delta U_{\tau}, \quad (11.47)$$

where  $\tau$  indexes the time step,  $\delta$  is the step size, and for notational simplicity, we continue to use  $U_{\tau}$  to denote the noise term, but here  $U_{\tau} \sim \mathcal{N}(0, I_d)$ .

We take the derivative of  $\log P(Z, \mathbf{I}; \theta)$  in (11.47) because this derivative is the same as the derivative of the log-posterior  $\log P_G(Z|\mathbf{I}; \theta)$ , since  $P(Z|\mathbf{I}; \theta)$  is proportional to  $P(Z, \mathbf{I}; \theta)$  as a function of  $Z$ .

To explain Langevin dynamics, its continuous time version for sampling  $\pi(Z) \propto \exp[-\mathcal{E}(Z)]$  is  $Z_{t+\Delta t} = Z_t - \Delta t \mathcal{E}'(Z_t)/2 + \sqrt{\Delta t} U_t$ . The dynamics has  $\pi$  as its stationary distribution, because it can be shown that for any well-behaved testing function  $h$ , if  $Z_t \sim \pi$ , then  $\mathcal{E}[h(Z_{t+\Delta t})] - \mathcal{E}[h(Z_t)] \rightarrow 0$ , as  $\Delta t \rightarrow 0$ , so that  $Z_{t+\Delta t} \sim \pi$ . Alternatively, given  $Z_t = Z$ , suppose  $Z_{t+\Delta t} \sim K(\mathbf{I}, y)$ , then  $[\pi(y)K(y, Z)]/[\pi(Z)K(Z, y)] \rightarrow 1$  as  $\Delta t \rightarrow 0$ .

The Langevin inference solves a  $\ell_2$  penalized non-linear least squares problem so that  $Z_i$  can reconstruct  $\mathbf{I}_i$  given the current  $W$ . The Langevin inference process performs explaining-away reasoning, where the latent factors in  $Z$  compete with each other to explain the current residual  $\mathbf{I} - g(Z; \theta)$ .

The stochastic gradient algorithm of [254] can be used for learning, where in each iteration, for each  $Z_i$ , only a single copy of  $Z_i$  is sampled from  $p(Z_i|\mathbf{I}_i, \theta)$  by running a finite number of steps of Langevin



dynamics starting from the current value of  $Z_i$ , i.e., the warm start. With  $Z_i$  sampled from  $P(Z_i | \mathbf{I}_i, \theta)$  for each observation  $\mathbf{I}_i$  by the Langevin inference process, the Monte Carlo approximation to  $L'(\theta)$  is

$$\begin{aligned} L'(\theta) &\approx \frac{1}{n} \sum_{i=1}^n \frac{\partial}{\partial W} \log P(Z_i, \mathbf{I}_i; \theta) \\ &= \frac{1}{n} \sum_{i=1}^n \frac{1}{\sigma^2} (\mathbf{I}_i - g(Z_i; \theta)) \frac{\partial}{\partial W} g(Z_i; \theta). \end{aligned} \quad (11.48)$$

The updating of  $W$  solves a non-linear regression problem so that the learned  $W$  enables better reconstruction of  $\mathbf{I}_i$  by the inferred  $Z_i$ . Given the inferred  $Z_i$ , the learning of  $W$  is a supervised learning problem [44].

**Input:**

- (1) training examples  $\{\mathbf{I}_i, i = 1, \dots, n\}$
- (2) number of Langevin steps  $l$
- (3) number of learning iterations  $T$

**Output:**

- (1) estimated parameters  $W$
- (2) inferred latent factors  $\{Z_i, i = 1, \dots, n\}$

- 1: Let  $t \leftarrow 0$ , initialize  $W$ .
- 2: Initialize  $Z_i, i = 1, \dots, n$ .
- 3: **repeat**
- 4:   **Step G1 Langevin inference:** For each  $i$ , run  $l$  steps of Langevin dynamics to update  $Z_i$ , i.e., starting from the current  $Z_i$ , each step follows equation (11.47).
- 5:   **Step G2 reconstruction:** Update  $W^{(t+1)} = W^{(t)} + \gamma_t L'(W^{(t)})$ , with learning rate  $\gamma_t$ , where  $L'(W^{(t)})$  is computed according to equation (11.48).
- 6:   Let  $t \leftarrow t + 1$
- 7: **until**  $t = T$

**Algorithm 7:** Algorithm G

Algorithm 7 [85] describes the training algorithm that iterates the following two steps: (1) Inference step: update  $Z_i$  by running  $l$  steps of gradient descent. (2) Learning step: update  $W$  by one step of gradient descent. Step G1 needs to compute  $\frac{\partial}{\partial Z} g(Z; \theta)$ . Step G2 needs to compute  $\frac{\partial}{\partial W} g(Z; \theta)$ . The computations of both derivatives can be powered by back-propagation, and the computations of the two derivatives share most of their steps in the chain rule computations.

Algorithm 7 is a stochastic approximation or stochastic gradient algorithm that converges to the maximum likelihood estimate [254]. If the Gaussian noise  $U_\tau$  in the Langevin dynamics (11.47) is removed, then the above algorithm becomes the alternating gradient descent algorithm. It is possible to update both  $W$  and  $\{Z_i\}$  simultaneously by joint gradient descent.

Both the inferential back-propagation and the learning back-propagation are guided by the residual  $\mathbf{I}_i - f(Z_i; \theta)$ . The inferential back-propagation is based on  $\partial f(Z; \theta) / \partial Z$ , whereas the learning back-propagation is based on  $\partial f(Z; \theta) / \partial W$ . Both gradients can be efficiently computed by back-propagation. The computations of the two gradients share most of their steps.

In Algorithm 7, the Langevin dynamics samples from a gradually changing posterior distribution  $p(Z_i | \mathbf{I}_i, \theta)$  because  $W$  keeps changing. The updating of both  $Z_i$  and  $W$  collaborate to reduce the reconstruction error  $\|\mathbf{I}_i - f(Z_i; \theta)\|^2$ . The parameter  $\sigma^2$  plays the role of annealing or tempering in Langevin sampling. If  $\sigma^2$

is very large, then the posterior is close to the prior  $N(0, I_d)$ . If  $\sigma^2$  is very small, then the posterior may be multi-modal, but the evolving energy landscape of  $p(Z_i|\mathbf{I}_i, \theta)$  may help alleviate the trapping of the local modes. In practice, we tune the value of  $\sigma^2$  instead of estimating it. The Langevin dynamics can be extended to Hamiltonian Monte Carlo [164] or more sophisticated versions [73].



Figure 11.21: Reprinted with permission from [85]. Modeling texture patterns. For each example, *Left*: the  $224 \times 224$  observed image. *Right*: the  $448 \times 448$  generated image.

Figure 11.21 illustrates the results of modeling textures where we learn a separate model from each texture image. The factors  $Z$  at the top layer form a  $\sqrt{d} \times \sqrt{d}$  image, with each pixel following  $N(0, 1)$  independently. The  $\sqrt{d} \times \sqrt{d}$  image  $Z$  is then transformed to  $\mathbf{I}$  by the top-down ConvNet. In order to obtain the synthesized image, we randomly sample a  $7 \times 7$  image  $Z$  from  $N(0, I)$ , and then expand the learned network  $W$  to generate the  $448 \times 448$  synthesized image  $f(Z; \theta)$ .

#### 11.2.4 EM, Density Mapping, and Density Shifting

Suppose the training data  $\{\mathbf{I}_i, i = 1, \dots, n\}$  come from a data distribution  $P_{\text{data}}(\mathbf{I})$ . To understand how the alternating back-propagation algorithm or its EM idealization maps the prior distribution of the latent factors  $p(Z)$  to the data distribution  $P_{\text{data}}(\mathbf{I})$  by the learned  $g(Z; \theta)$ , we define

$$\begin{aligned} P_{\text{data}}(Z, \mathbf{I}; \theta) &= P_{\text{data}}(\mathbf{I})p(Z|\mathbf{I}, \theta) \\ &= P_{\text{data}}(Z; \theta)P_{\text{data}}(\mathbf{I}|Z, \theta), \end{aligned} \quad (11.49)$$

where  $P_{\text{data}}(Z; \theta) = \int p(Z|\mathbf{I}, \theta)P_{\text{data}}(\mathbf{I})d\mathbf{I}$  is obtained by averaging the posteriors  $p(Z|\mathbf{I}; \theta)$  over the observed data  $\mathbf{I} \sim P_{\text{data}}$ . That is,  $P_{\text{data}}(Z; \theta)$  can be considered the data prior. The data prior  $P_{\text{data}}(Z; \theta)$  is close to the true prior  $p(Z)$  in the sense that

$$\begin{aligned} \text{KL}(P_{\text{data}}(Z; \theta)|p(Z)) &\leq \text{KL}(P_{\text{data}}(\mathbf{I})|p(\mathbf{I}; \theta)) \\ &= \text{KL}(P_{\text{data}}(Z, \mathbf{I}; \theta)|p(Z, \mathbf{I}; \theta)). \end{aligned} \quad (11.50)$$

The difference between the two sides of (11.50) is  $\text{KL}(P_{\text{data}}(\mathbf{I}|Z, \theta) | p(\mathbf{I}|Z, \theta))$ . The right hand side of (11.50) is minimized at the maximum likelihood estimate  $\hat{W}$ , hence the data prior  $P_{\text{data}}(Z; \hat{W})$  at  $\hat{W}$  should be especially close to the true prior  $p(Z)$ . In other words, at  $\hat{W}$ , the posteriors  $p(Z|\mathbf{I}, \hat{W})$  of all the data points  $\mathbf{I} \sim P_{\text{data}}$  tend to pave the true prior  $p(Z)$ .

From Rubin's multiple imputation point of view [190] of the EM algorithm, the E-step of EM infers  $Z_i^{(m)} \sim p(Z_i|\mathbf{I}_i, W_t)$  for  $m = 1, \dots, M$ , where  $M$  is the number of multiple imputations or multiple guesses of  $Z_i$ . The multiple guesses account for the uncertainty in inferring  $Z_i$  from  $\mathbf{I}_i$ . The M-step of EM maximizes  $Q(\theta) = \sum_{i=1}^n \sum_{m=1}^M \log p(\mathbf{I}_i, Z_i^{(m)}; \theta)$  to obtain  $W_{t+1}$ . For each data point  $\mathbf{I}_i$ ,  $W_{t+1}$  seeks to reconstruct  $\mathbf{I}_i$  by  $g(Z; W)$  from the inferred latent factors  $\{Z_i^{(m)}, m = 1, \dots, M\}$ . In other words, the M-step seeks to map  $\{Z_i^{(m)}\}$  to  $\mathbf{I}_i$ . Pooling over all  $i = 1, \dots, n$ ,  $\{Z_i^{(m)}, \forall i, m\} \sim P_{\text{data}}(Z; W_t)$ , hence the M-step seeks to map  $P_{\text{data}}(Z; W_t)$  to the data distribution  $P_{\text{data}}(\mathbf{I})$ . Of course the mapping from  $\{Z_i^{(m)}\}$  to  $\mathbf{I}_i$  cannot be exact. In fact,  $g(Z; \theta)$  maps  $\{Z_i^{(m)}\}$  to a  $d$ -dimensional patch around the  $D$ -dimensional  $\mathbf{I}_i$ . The local patches for all  $\{\mathbf{I}_i, \forall i\}$  patch up the  $d$ -dimensional manifold form by the  $D$ -dimensional observed examples and their interpolations. The EM algorithm is a process of density shifting, so that  $P_{\text{data}}(Z; \theta)$  shifts towards  $p(Z)$ , thus  $g(Z; \theta)$  maps  $p(Z)$  to  $P_{\text{data}}(\mathbf{I})$ .

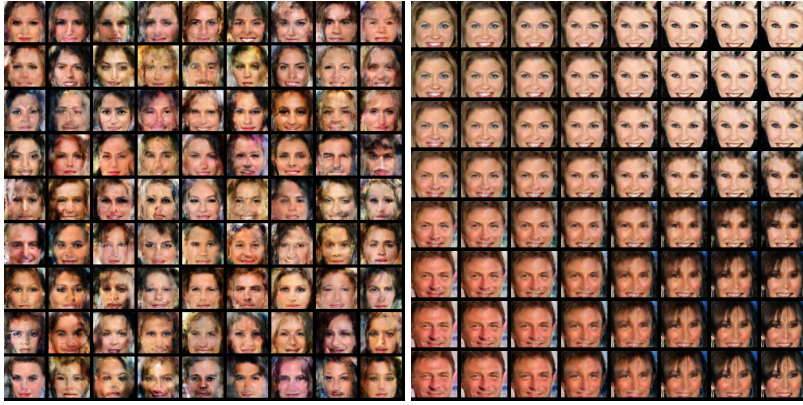


Figure 11.22: Reprinted with permission from [85]. Modeling object patterns. *Left*: each image generated by our method is obtained by first sampling  $Z \sim \mathcal{N}(0, I_{100})$  and then generating the image by  $g(Z; \theta)$  with the learned  $W$ . *Right*: interpolation. The images at the four corners are reconstructed from the inferred  $Z$  vectors of four images randomly selected from the training set. Each image in the middle is obtained by first interpolating the  $Z$  vectors of the four corner images, and then generating the image by  $g(Z; \theta)$ .

Figure 11.22 illustrates the learned  $D$ -dimensional manifold. We learn a model where  $Z = (z_1, z_2, \dots, z_d)$  has  $d = 100$  components from 1000 face images randomly selected from the CelebA dataset [140]. The left panel of Figure 11.22 displays the images generated by the learned model. The right panel displays the interpolation results. The images at the four corners are generated by the  $Z$  vectors of four images randomly selected from the training set. The images in the middle are obtained by first interpolating the  $Z$ 's of the four corner images using the sphere interpolation [41] and then generating the images by the learned ConvNet.

### 11.2.5 Extracting Appearance and Geometry, Nonlinear Generalization of the AAM Model

Active Appearance Models (AAM) [27, 123] uses a linear model for jointly capturing the appearance and geometric variation in an image. For the face images, the appearance information mainly includes colors,

illuminations and identities, while the geometric information mainly includes the shapes and viewing angles. Given a set of landmark points, the AAM model can learn the eigen vectors from these landmarks to extract the geometric information orthogonally. With the known landmark points, the faces can also be aligned into a canonical shape and view, by warping the faces with mean landmarks. Under this canonical geometric state, the appearance information can be extracted orthogonally by the principal component analysis.

Can we extract the appearance and geometric knowledge from the images without any landmarks or any supervised information? Moreover, can we disentangle the appearance and geometric information in a purely unsupervised method? The deformable generator model can solve this problem.

The deformable generator model [249, 250] disentangle the appearance and geometric information from images into two independent latent vectors. The model contains two generator networks: one appearance generator and one geometric generator. The appearance generator produces the appearance information, including color, illumination, identity or category, of an image. The geometric generator produces a displacement of the coordinates of each pixel and performs geometric warping, such as stretching and rotation, on the appearance generator to obtain the final synthesized image, as shown in Figure 11.23. The geometric operation only modifies the positions of pixels in an image without changing their colors and illumination. Therefore, the color and illumination information and the geometric information are naturally disentangled by the geometric generator and the appearance generator in the model.

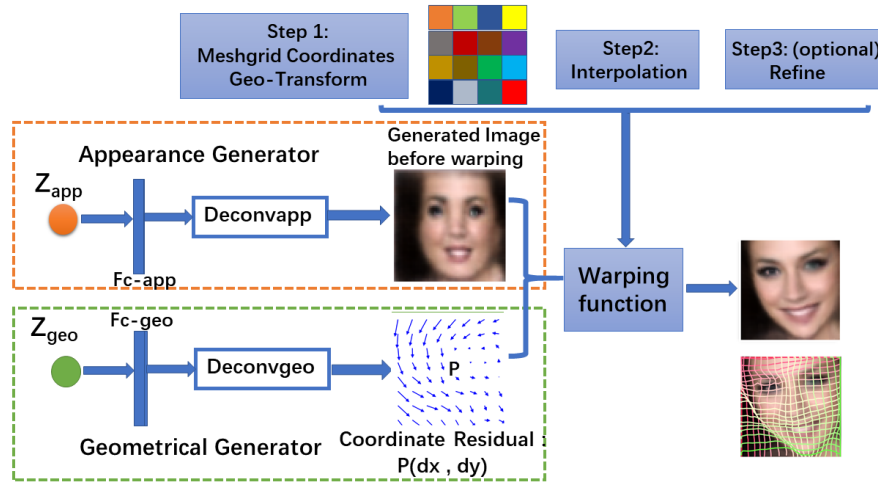


Figure 11.23: Reprinted with permission from [249]. An illustration of the proposed model. The model contains two generator networks: one appearance generator and one geometric generator. The two generators are connected with a warping function to produce the final image. The warping function includes a geometric transformation operation for image coordinates and a differentiable interpolation operation. The refining operation is optional for improving the warping function.

The model can be expressed as

$$\begin{aligned} \mathbf{I} &= G(Z^a, Z^g; \theta) + \epsilon \\ &= F_w(g_a(Z^a; \theta_a), g_g(Z^g; \theta_g)) + \epsilon \end{aligned} \quad (11.51)$$

where  $Z^a \sim N(0, I_{d_a})$ ,  $Z^g \sim N(0, I_{d_g})$ , and  $\epsilon \sim N(0, \sigma^2 I_D)$  ( $D = D_x \times D_y \times 3$ ) are independent.  $F_w$  is the warping function, which uses the deformation field generated by the geometric generator  $g_g(Z^g; \theta_g)$  to warp the image generated by the appearance generator  $g_a(Z^a; \theta_a)$  to synthesize the final output image  $\mathbf{I}$ .

This deformable generator model can be learned by extending the alternating back-propagation algorithm for two latent vectors. Specifically, the proposed model can be trained by maximizing the log-likelihood on the training dataset  $\{\mathbf{I}_i, i = 1, \dots, N\}$ ,

$$\begin{aligned} L(\theta) &= \frac{1}{N} \sum_{i=1}^N \log p(\mathbf{I}_i; \theta) \\ &= \frac{1}{N} \sum_{i=1}^N \log \int p(X_i, Z_i^a, Z_i^g; \theta) dZ_i^a dZ_i^g, \end{aligned} \quad (11.52)$$

where we integrate out the uncertainties of  $Z_i^a$  and  $Z_i^g$  in the complete-data log-likelihood to get the observed-data log-likelihood.

We can evaluate the gradient of  $L(\theta)$  by the following well-known result, which is related to the EM

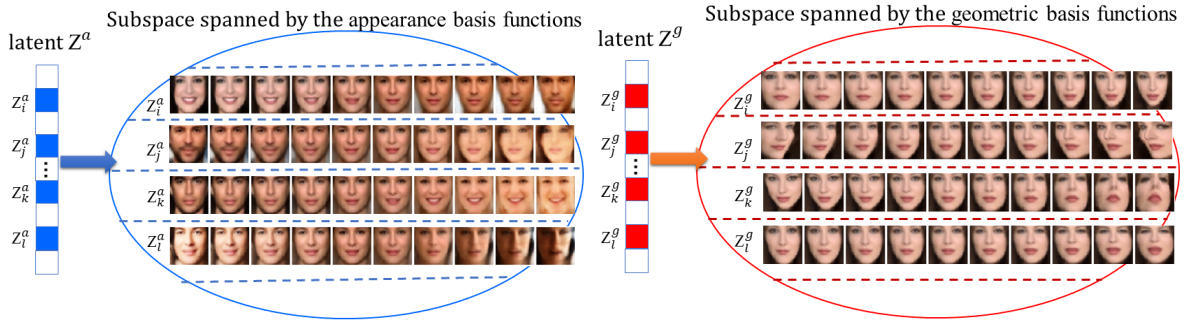


Figure 11.24: Reprinted with permission from [249]. Extracting appearance and geometric basis functions. *Left:* Each dimension of the appearance latent vector encodes appearance information such as color, illumination, and gender. In the first line, from left to right, the color of the background varies from black to white, and the gender changes from a woman to a man. In the second line, the mustache of the man becomes thicker when the corresponding dimension of  $Z^a$  approaches zero, and the hair of the woman becomes denser when the corresponding dimension of  $Z^a$  increases. In the third line, from left to right, the skin color changes from dark to white. In the fourth line, from left to right, the illumination lighting changes from the left side of the face to the right side of the face. *Right:* Each dimension of the geometric latent vector encodes fundamental geometric information such as shape and viewing angle. In the first line, the shape of the face changes from fat to thin from left to the right. In the second line, the pose of the face varies from left to right. In the third line, from left to right, the vertical tilt of the face varies from downward to upward. In the fourth line, the face width changes from stretched to cramped. The deformable generator model is trained on the 10,000 face images from the CelebA data set, which are cropped to  $64 \times 64$  pixels, and the faces in the training data have a wide and diverse variety of colors, illuminations, identities, viewing angles, shapes, and expressions.

algorithm:

$$\begin{aligned} &\frac{\partial}{\partial \theta} \log p(\mathbf{I}; \theta) \\ &= \frac{1}{p(\mathbf{I}; \theta)} \frac{\partial}{\partial \theta} \int p(\mathbf{I}, Z^a, Z^g) dZ^a dZ^g \\ &= \mathcal{E}_{p(Z^a, Z^g | \mathbf{I}; \theta)} \left[ \frac{\partial}{\partial \theta} \log p(\mathbf{I}, Z^a, Z^g; \theta) \right] \end{aligned} \quad (11.53)$$



Since the expectation in Eq.(11.53) is usually analytically intractable, we employ Langevin dynamics to draw samples from the posterior distribution  $p(Z^a, Z^g | \mathbf{I}; \theta)$  and compute the Monte Carlo average to estimate the expectation term. For each observation  $\mathbf{I}$ , the latent vectors  $Z^a$  and  $Z^g$  can be sampled from  $p(Z^a, Z^g | \mathbf{I}; \theta)$  alternately by Langevin dynamics: we fix  $Z^g$  and sample  $Z^a$  from  $p(Z^a | \mathbf{I}; Z^g, \theta) \propto p(\mathbf{I}, Z^a; Z^g, \theta)$ , and then fix  $Z^a$  and sample  $Z^g$  from  $p(Z^g | \mathbf{I}; Z^a, \theta) \propto p(\mathbf{I}, Z^g; Z^a, \theta)$ . At each sampling step, the latent vectors are updated as follows:

$$\begin{aligned} Z_{\tau+1}^a &= Z_{\tau}^a + \frac{\delta^2}{2} \frac{\partial}{\partial Z^a} \log p(\mathbf{I}, Z_{\tau}^a; Z_{\tau}^g, \theta) + \delta \mathcal{E}_{\tau}^a \\ Z_{\tau+1}^g &= Z_{\tau}^g + \frac{\delta^2}{2} \frac{\partial}{\partial Z^g} \log p(\mathbf{I}, Z_{\tau}^g; Z_{\tau}^a, \theta) + \delta \mathcal{E}_{\tau}^g \end{aligned} \quad (11.54)$$

where  $\tau$  is the number of steps in the Langevin sampling,  $\mathcal{E}_{\tau}^a, \mathcal{E}_{\tau}^g$  are independent standard Gaussian noise to prevent the sampling from being trapped in local modes, and  $\delta$  is the step size. The complete-data log-likelihood can be evaluated by

$$\begin{aligned} \log p(\mathbf{I}, Z^a; Z^g, \theta) &= \log [p(Z^a)p(\mathbf{I} | Z^a, Z^g, \theta)] \\ &= -\frac{1}{2\sigma^2} \|\mathbf{I} - F(Z^a, Z^g; \theta)\|^2 - \frac{1}{2} \|Z^a\|^2 + C_1 \\ \log p(\mathbf{I}, Z^g; Z^a, \theta) &= \log [p(Z^g)p(\mathbf{I} | Z^a, Z^g, \theta)] \\ &= -\frac{1}{2\sigma^2} \|\mathbf{I} - F(Z^a, Z^g; \theta)\|^2 - \frac{1}{2} \|Z^g\|^2 + C_2 \end{aligned} \quad (11.55)$$

where  $C_1$  and  $C_2$  are normalizing constants. It can be shown that, given sufficient sampling steps, the sampled  $Z^a$  and  $Z^g$  follow their joint posterior distribution.

After training the model, two sets of basis functions related to the appearance and geometric latent vectors can be learned. The appearance and the geometric latent factors can be interpreted as the projection or reconstruction coefficients along the direction of the corresponding appearance and geometric basis functions. Representative basis functions from these two sets are shown in Figure 11.24. The appearance basis function is obtained by setting the geometric latent vector to be zero, and only keeping one dimension of the appearance latent vector to be non-zeros. Then, feeding these groups of appearance and geometric latent vectors to the top-down deformable generator to produce the corresponding basis functions. To study the appearance latent vector, each time vary one dimension of the appearance variable  $Z^a$  from  $[-\gamma, \gamma]$  with a uniform step  $\frac{2\gamma}{10}$ , while holding the other dimensions of  $Z^a$  at zero. And vice versa for the appearance basis function and latent vector.

The abstracted geometric knowledge can be conveniently transferred to facilitate downstream AI tasks. For the unseen images, Figure 11.25, we can first infer their appearance and geometric latent vectors, then the geometric knowledge can be conveniently transferred by recombining the inferred geometric latent vector from the target image with the inferred appearance latent vector from the source image.

For high resolution images, we train the deformable generator with 40K faces from FFHQ [116], which are cropped to  $256 \times 256$  pixels. Similar results corresponding to Figure 11.24 and Figure 11.25 for the generated basis functions of appearance and geometry, as well as the transferring and recombining the vectors are demonstrated at Figure 11.26 and Figure 11.27. Since it is well known that the squared Euclidean distance induced from the MLE loss often yields blurry reconstruction and generation results, we recruit adversarial training which includes the deformable generator as an actor and a discriminator which acts as a critic. We extend the MLE loss in Eq. (11.52) with the adversarial loss, i.e., the non-saturating loss [74] with  $R_1$  regularization [156],

$$\min_{\theta} \max_{\phi} T(\theta, \phi),$$

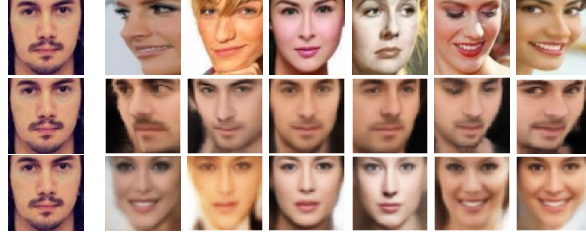


Figure 11.25: Reprinted with permission from [249]. Transferring and recombining geometric and appearance vectors. The first row shows seven faces from the CelebA data set. The second row shows the faces generated by transferring and recombining the second through seventh faces' geometric vectors  $z_1$  with the first face's appearance vector  $z_2$  in the first row. The third row shows the faces generated by transferring and recombining the second through seventh faces' appearance vectors  $z_2$  with the first face's geometric vector  $z_1$  in the first row. The deformable generator model is trained on the 10,000 face images from the CelebA data set, which are cropped to  $64 \times 64$  pixels, and the faces in the training data have a wide and diverse variety of colors, illuminations, identities, viewing angles, shapes, and expressions.

$$T(\theta, \phi) = -\lambda_1 L(\theta) + \mathcal{E}_{Z^a, Z^g} [\log(1 - D(G(Z^a, Z^g; \theta); \phi))] \quad (11.56)$$

$$+ \mathcal{E}_{\mathbf{I}} [\log D(X; \phi)] + \frac{\lambda_2}{2} \mathcal{E}_{\mathbf{I}} [\| \nabla D(\mathbf{I}; \phi) \|^2],$$

where  $L(\theta)$  and  $G(Z^a, Z^g; \theta)$  is defined in Eq.(11.51) and (11.52).  $D(\cdot; \phi)$  is the discriminator.  $\lambda_1 = 0.001$  and  $\lambda_2 = 10$ .

### 11.2.6 Sparse Generator Model

The sparsity principle has played a fundamental role in high-dimensional statistics, machine learning, and signal processing. In particular, sparse coding [173] is an important principle for understanding the visual cortex. By imposing sparsity constraints on the coefficients of a linear generative model, [174] learn Gabor-like wavelets that resemble the neurons in the primary visual cortex (V1) from natural image patches.

However, developing a top-down sparse coding model that can generate (in addition to merely reconstructing) realistic natural image patterns has proven to be a difficult task. The model of [174] assumes that the coefficients of the linear model follow a sparse distribution. However, it is difficult to develop a realistic model for the sophisticated sparse patterns, which involves a selection process that selects which coefficients are active, in addition to generating the values of the active coefficients. The commonly used independent spike and slab model [108] can hardly generate realistic images. Moreover, even if we do have such a selection model as a prior model for the coefficients, fitting the model to each training example involves a non-trivial inference process to identify the active coefficients and estimate their values. Because of the modeling and computing difficulties, we still do not have a realistic top-down generative model that has the linear sparse coding model at the lowest layer or incorporates the sparse coding principle across its layers.

#### Basis functions and sparse representations

Suppose  $\mathbf{I}(x)$  denotes an image defined on the spatial domain  $\mathbb{D}$ , where  $x = (x_1, x_2)$  denotes a two-dimensional vector that indexes the coordinates of pixels.  $\mathbf{I}(x)$  can be treated as a two-dimensional function defined on  $\mathbb{D}$ .  $\mathbf{I}(x)$  can also be treated as a vector if we fix an ordering for the pixels. Suppose  $D = |\mathbb{D}|$  counts the number of pixels in  $\mathbb{D}$ , then  $D$  is the dimensionality of the vector  $\mathbf{I}$ .

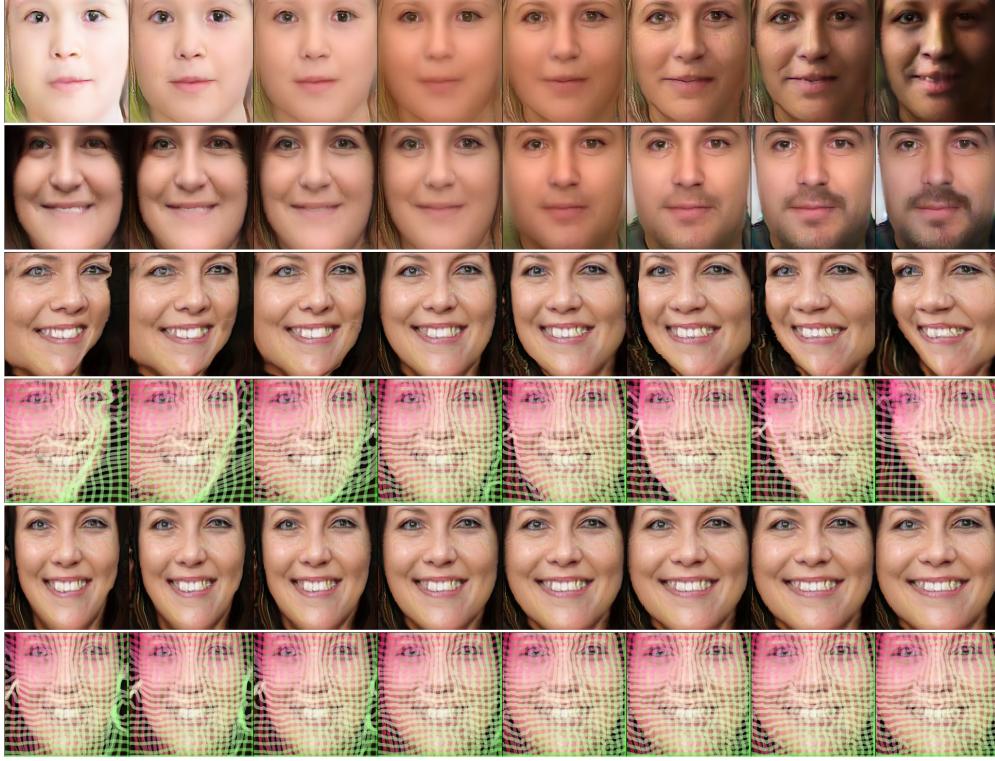


Figure 11.26: Reprinted with permission from [249]. Extracting appearance and geometric basis functions. The typical appearance basis functions are shown in the first two rows. Each dimension of the appearance latent vector encodes appearance information such as color, illumination, and gender. In the first line, from left to right, the illumination changes from light to dark. In the second line, the color of the background varies from black to white, and the gender changes from a woman to a man. The Representative geometric basis functions are demonstrated in the rest of four rows with the corresponding deformable grids overlaid. Each dimension of the geometric latent vector encodes fundamental geometric information such as viewing angle and shape. In the third line, the pose of the face varies from left to right. In the fifth line, the shape of the face changes from thin to fat from left to the right. The deformable generator model is trained on the 40,000 face images randomly selected from FFHQ dataset [116]. The training images are cropped to  $256 \times 256$  pixels, and the faces have different colors, illuminations, identities, viewing angles, shapes, and expressions.

A linear basis function (or basis vector) is a local image patch, which is utilized to represent image intensities. Let  $\{W_k(x), k = 1, \dots, K\}$  be a set of prototype basis functions, e.g., wavelets. Suppose that each  $W_k$  is supported on a local domain  $\mathbb{S}$  centered at the origin,  $W_k$  can be shift or translate spatially to a position  $x_t$  to get a translated copy of  $W_k$  as  $W_{k,x_t} = W_k(x - x_t)$ .  $W_{k,x_t}(x)$  can be treated as a locally supported function defined on  $x \in \mathbb{D}$ .  $W_{k,x_t}(x)$  can also be treated as a vector of the same dimensionality as  $\mathbf{I}$ . The basis functions are about top-down linear representation (top-down means from the coefficients to the image) of an image  $\mathbf{I}$  by

$$\mathbf{I}(x) = \sum_{k,x_t} s_{k,x_t} W_{k,x_t}(x) + \epsilon(x)$$





Figure 11.27: Reprinted with permission from [249]. Transferring and recombining the abstracted geometric and appearance knowledge. The first row shows 8 unseen faces from FFHQ. The second row shows the generated faces by transferring and recombining 2th-8th faces’ geometric vectors with the first face’s appearance vector. The third row shows the generated faces by recombining the 2th-8th faces’ appearance vectors with the first face’s geometric vector in the first row. The deformable generator model is trained on the 40,000 face images randomly selected from FFHQ dataset [116]. The training images are cropped to  $256 \times 256$  pixels, and the faces have different colors, illuminations, identities, viewing angles, shapes, and expressions.

where  $S_{k,x_t}$  are the coefficients, and  $\epsilon(x)$  is the residual image. When most of the  $s_{k,x_t}$  are equal to zero, the resulting representation is also called sparse coding.

The traditional one-layer sparse coding model generalizes the prior distribution of the coefficients vector to the signal in factor analysis from a Gaussian noise vector to a sparse vector. And typically comprise two steps: (1) Learning a dictionary of basis functions  $\{W_{k,x_t}\}$ , where  $k$  index a finite collection of prototype functions  $\{W_k, k = 1, \dots, K\}$ , that  $W_{k,x_t}$  is a spatially translated copy of  $W_k$ . (2) Inferring the latent factor ( $S$ ),

$$W^*, S^* = \arg \min_{W, S} \| \mathbf{I} - WS \|_F^2 \quad \| s_{x_t} \|_0 \leq K$$

K-SVD is a typical algorithm for learning the dictionary of basis functions  $W_k$ , while convex relaxation  $L_1$  minimization or greedy methods such as matching pursuit (OMP) can be employed to infer the sparsification  $s_{k,x_t}$ , i.e., the selection of the basis functions. We will develop a hierarchical top-down sparse coding model that can generate (in addition to merely reconstructing) the realistic natural image patterns [251, 252].

### Inducing unified hierarchical sparse coding model from sparsely connected generator

Unlike the sparse coding model, the generator network is a dense model without sparsity. In this chapter, we try to unify the sparse coding model and the top-down generator network by introducing a deep sparsely generator to connect the sparsity principle to the compositionality principle, or more specifically, we propose to learn hierarchical compositional AND-OR model for interpretable image synthesis by sparsifying the generator network. The proposed method adopts the scene-objects-parts-subparts-primitives hierarchy in image representation. In the language of AND-OR grammars [260], a scene has different types (i.e., OR) each of which consists of a number of objects (i.e., AND). This can be recursively formulated across the scene-objects-parts-subparts hierarchy and is terminated at the primitive level (e.g., wavelet-like basis).

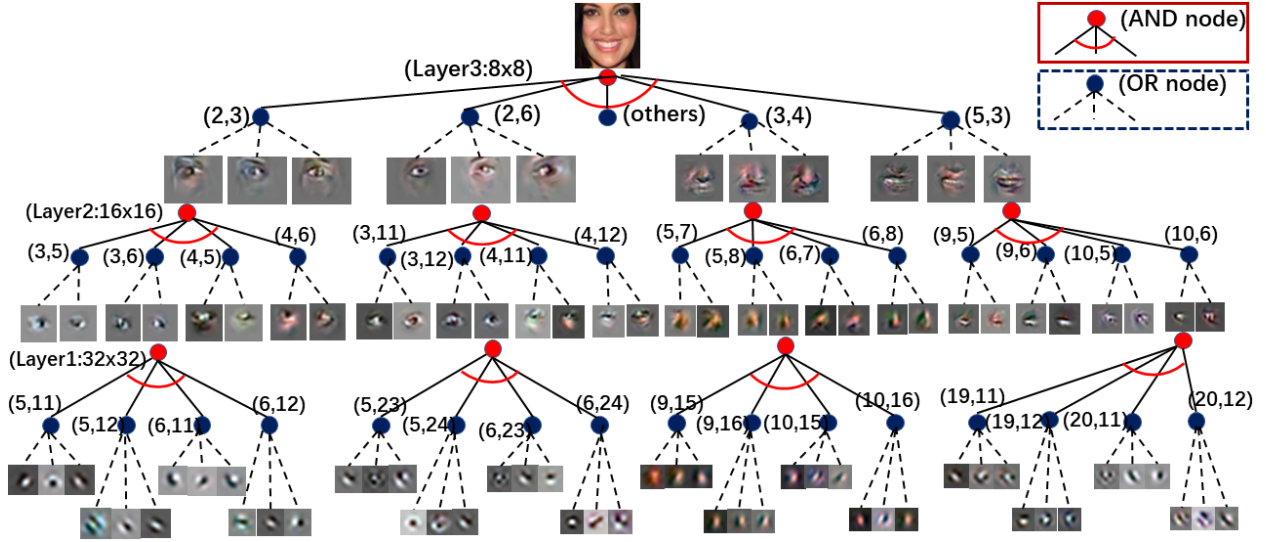


Figure 11.28: Reprinted with permission from [251]. An example of AND-OR tree learned for face synthesis at a resolution of  $64 \times 64$ . For clarity, we only show 3 layers (out of the total 5 layers). The AND-OR tree is illustrated from Layer 3 (the part/composite part level) to Layer 1 (the primitive level). The spatial resolution of the feature map at Layer 3 is  $8 \times 8$ . The entire grid is interpreted by an AND-node. Each position  $(x, y)$  in the  $8 \times 8$  grid is interpreted by an OR-node such as the eye node at the positions (2, 3) and (2, 6). Similarly, Layer 2 and Layer 1 can be interpreted w.r.t. the AND-OR compositions. The activated basis functions have semantically meaningful interpretations at Layer 3 and Layer 2. Layer 1 shows the learned primitives covering the classic Gabor-like wavelets and the blob-like primitives. See text for detail. Best viewed in color.

The deep sparsely generator is explainable since the internal image generation process can be explicitly unfolded through meaningful basis functions at different layers which are learned via end-to-end training and which conceptually reflect the hierarchy of scene-objects-parts-subparts-primitives. Figure 11.28 shows an example of the AND-OR tree learned from scratch for explaining the generation of a face image.

To accomplish this, we learn a generator network that consists of the following two components: (i) Each layer of the hierarchy is represented by an over-complete set of convolutional basis functions. Off-the-shelf convolutional neural architectures are exploited to implement the hierarchy. (ii) Sparsity-inducing constraints are introduced in end-to-end training, which induces a sparsely activated and sparsely connected AND-OR model from the initially densely connected generator network. A straightforward sparsity-inducing constraint is utilized, that is to only allow the top- $k$  basis functions to be activated at each layer (where  $k$  is a hyper-parameter). The learned basis functions are also capable of image reconstruction to explain the input images. The top-down sparse coding generator network can learn interpretable dictionaries at multiple layers, moreover, learn the compositional structures and primitives automatically from data.

From the viewpoint of top-down generative learning in image synthesis, we start with a  $d$ -dimensional latent code vector  $Z = (z_i, i = 1, \dots, d)$  consisting of  $d$  latent factors. We usually assume  $Z \sim \mathcal{N}(0, I_d)$ , where  $I_d$  denotes the  $d$ -dimensional identity matrix. In GANs and VAE, generator networks are used to implement the highly non-linear mapping from a latent code vector  $Z$  to a synthesized image, denoted by  $Y$  which lies in a  $D$ -dimensional image space (i.e.,  $D$  equals the product of the spatial dimensions, width and height of an image, and the number of chromatic channels such as 3 for RGB images). The generator

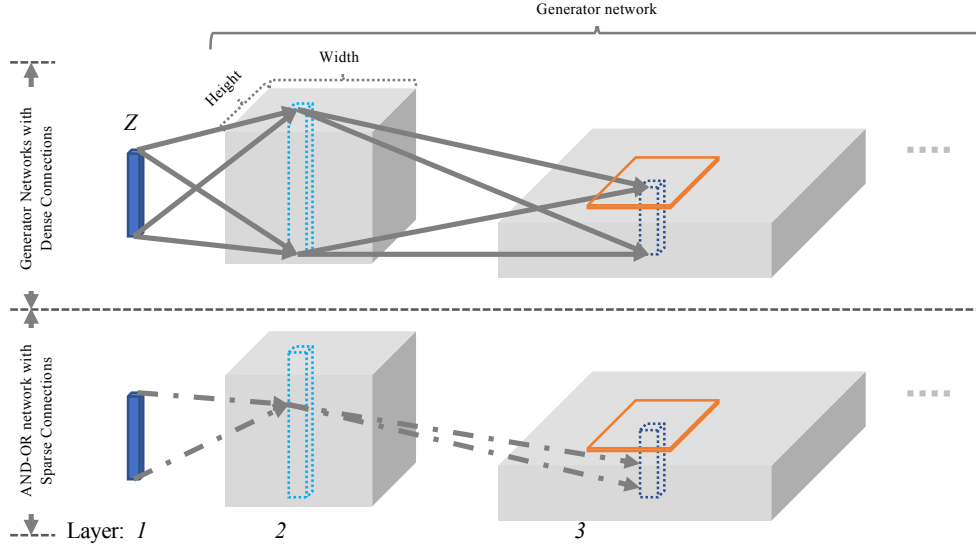


Figure 11.29: Reprinted with permission from [251]. *Top*: traditional generator networks with dense connections (solid arrows) between consecutive layers, which are widely used in GANs and VAE. *Bottom*: the proposed AND-OR models with sparse connections (dashed arrows). See text for detail.

network is thus seen as the non-linear extension of factor analysis. The model has the following form:

$$\begin{aligned} \mathbf{I} &= g(\mathbf{Z}; \Theta) + \epsilon, \\ \mathbf{Z} &\sim \mathcal{N}(0, \mathbf{I}_d), \quad \epsilon \sim \mathcal{N}(0, \sigma^2 \mathbf{I}_D), \quad d < D, \end{aligned} \quad (11.57)$$

where  $\epsilon$  is the observational errors assumed to be Gaussian white noises,  $g(\cdot)$  represents the generator network and  $\Theta$  collects parameters from all layers.

As illustrated at the top of Figure 11.29, dense connections between consecutive layers are learned in the vanilla generator network, which we think is the main drawback that hinders explicit model interpretability. We explore and exploit the AND-OR compositionality in image synthesis by learning to rewire the connections sparsely and to unfold the internal image generation process in an interpretable way, as illustrated at the bottom of Figure 11.29.

### The induced AND-OR model

Without loss of generality, consider a simple hierarchy of object(O)-part(P)-primitive/basis(B) that generates RGB images. Start with the latent code vector  $\mathbf{Z} \in \mathbb{R}^d$ , we have,

$$\text{Hierarchy: } \mathbf{Z} \rightarrow \mathbf{O} \rightarrow \mathbf{P} \rightarrow \mathbf{B} \rightarrow \mathbf{Y}, \quad (11.58)$$

$$\text{Layer Index: } 1, \quad 2, \quad 3, \quad 4, \quad 5. \quad (11.59)$$

For example, Figure 11.29 illustrates the computing flow from Layer 1 to Layer 3.

The symbol  $O$  in the hierarchy is grounded in an internal  $(H_2 \times W_2 \times d_2)$ -dimensional space, which can be treated as  $H_2 \times W_2$   $d_2$ -dimensional vectors when instantiated. Similarly, the symbols  $P$  and  $B$  will be instantiated as  $H_3 \times W_3$   $d_3$ -dimensional vectors and  $H_4 \times W_4$   $d_4$ -dimensional vectors respectively.  $Y$  is a generated RGB image of size  $H_5 \times W_5 \times 3$ .

To better show how to facilitate the sparse connections to emerge from the dense ones, we look at the computing flow using the lens of vector-matrix multiplication. In the vanilla generator network, consider a  $d_l$ -dimensional vector,  $U$  in Layer  $l$ , it connects to a set of  $d_{l+1}$ -dimensional vectors,  $V_i$ 's in Layer  $l+1$ . Let  $ch(U)$  be the set of indices of the vectors in Layer  $l+1$  which connect with  $U$  (i.e.,  $U$ 's child nodes). We have

$$V_i(U) = W_{d_{l+1} \times d_l} \cdot U + b, \quad i \in ch(U), \quad (11.60)$$

where  $V_i(U)$  means the contribution of  $U$  to  $V_i$  since there may be other vectors  $U'$  in Layer  $l$  connecting to  $V_i$  too.  $W_{d_{l+1} \times d_l}$  is the transformation matrix and  $b$  the bias vector. Consider Layer 1 to Layer 2 ( $Z \rightarrow O$ ),  $U = Z$  is connected with all vectors  $V_i$ 's in  $O$  with different  $W_{d_{l+1} \times d_l}$ 's and  $b$ 's. Consider Layer 2 to Layer 3 ( $O \rightarrow P$ ), convolution is usually used, so each  $U$  only connects to vectors  $V_i$ 's locally, and  $W_{d_{l+1} \times d_l}$ 's and  $b$ 's are shared among different  $U$ 's.

Denote by  $pr(V_i)$  the set of indices of vectors in Layer  $l$  connecting with  $V_i$ . In the vanilla generator network,

$$V_i = act\left(\sum_{j \in pr(V_i)} V_i(U_j)\right), \quad (11.61)$$

where  $act(\cdot)$  stands for activation function such as the ReLU function.

In the proposed method, we compute  $V_i$  by,

$$V_i = S\left(act\left(\sum_{j \in pr(V_i)} V_i(U_j)\right); k_{l+1}\right), \quad (11.62)$$

where  $S(\cdot; k_{l+1})$  is the sparsity-inducing function. From symbol  $Z$  to  $O$ , we apply the sparsity-inducing function along the  $d_{l+1}$  dimension and retain the top  $k_{l+1}$  out of  $d_{l+1}$  elements in the resulting vector in terms of the element values. In the subsequent layers, we apply it along the spatial domain across the  $d_{l+1}$  dimensions individually. By doing so, the resulting vectors at different locations will have different sparsity ratios. The  $k_l$ 's are hyper-parameters. We usually set  $d_l > d_{l+1}$  and  $k_l < k_{l+1}$ , that is, Layer  $l$  has higher sparsity degree than lower Layer  $l+1$ .

With sparsity-inducing functions, image synthesis is fundamentally changed in terms of representation. The internal generation process is also much easier to unfold. The AND-OR model then emerges from the vanilla dense connected generator network. We can rewrite Eqn. 11.57 as

$$\begin{aligned} \mathbf{I} &= g(Z; \Theta, \mathbf{k}) + \epsilon, \\ Z &\sim \mathcal{N}(0, I_d), \quad \epsilon \sim \mathcal{N}(0, \sigma^2 I_D), \quad d < D, \end{aligned} \quad (11.63)$$

where the sparsity hyper-parameters  $\mathbf{k} = \{k_l; l = 1, \dots, L\}$ . We summarize the proposed AND-OR model for image synthesis as follows.

**Layer 1 to Layer 2:**  $Z \rightarrow O$ . The latent code vector  $Z$  is represented by a root OR-node (non-terminal symbol),

$$Z \xrightarrow{\text{OR}} z_1 | z_2 | \dots | z_i | \dots, \quad z_i \stackrel{\text{i.i.d.}}{\sim} \mathcal{N}(0, I_d), \quad (11.64)$$

where  $a|b$  denotes OR switching between symbols  $a$  and  $b$  (i.e., instantiated latent code vectors that generate different object images).

Each instantiated latent code vector  $z_i$  is then mapped to an object instance AND-node  $O_i$ . The object instance AND-node  $O_i$  represents the object-part decomposition in the lattice  $\Lambda_2$  (of size  $H_2 \times W_2$ ). We have,

$$O_i \xrightarrow{\text{AND}} o_{i,1} \cdot o_{i,2} \cdot \dots \cdot o_{i,j} \cdot \dots \cdot o_{i,N_P}, \quad (11.65)$$

where  $a \cdot b$  represents the composition between symbols  $a$  and  $b$ .  $N_P$  is the number of part symbols. The object-part decomposition is usually done in the spatial domain. For example, if the support domain for  $o_i$ 's is  $4 \times 4$ , we will have at most 16 parts. We could use 4 parts if we further divide the  $4 \times 4$  domain into  $2 \times 2$  blocks.

Each  $o_{i,j}$  is then represented by an OR-node in the  $d_2$ -dimensional vector space indicating the sparse selection among  $\binom{d_2}{k_2}$  candidates. When instantiated, we have part AND-node  $o_{i,j}(k_2)$ .

**Layer 2 to Layer 3:**  $O \rightarrow P$ . Each part AND-node  $o_{i,j}(k_2)$  is decomposed into a number of  $M$  child part type OR-nodes,

$$o_{i,j}(k_3) \xrightarrow{\text{AND}} P_{i,j,1} \cdot P_{i,j,2} \cdots P_{i,j,M}, \quad (11.66)$$

where  $M$  is determined by the kernel size when convolution is used to compute Layer 3 from Layer 2.

Similarly, each part type OR-node  $P_{i,j,t}$  is grounded in the  $d_3$ -dimensional vector space indicating the sparse selection among  $\binom{d_3}{k_3}$  candidates. When instantiated, we have part-primitive AND-node. Then, the AND-OR is recursively formulated in the downstream layers. Now, let us look at Figure 11.29 again, for each instantiated  $z_i$ , we can follow the sparse connections and visualize the encountered kernel symbols (see Figure 11.28).

## Learning and inference

The proposed model can still utilize an off-the-shelf end-to-end learning framework since the sparsity-inducing functions do not change the formulation (Eqn. 11.63). Denote by  $\{\mathbf{I}_i, i = 1, \dots, N\}$  the training dataset consisting of  $N$  images (e.g., face images). The learning objective is to maximize the observed data log-likelihood

$$L(\Theta) = \frac{1}{N} \sum_{i=1}^N \log P(\mathbf{I}_i; \Theta, \mathbf{k}) = \frac{1}{N} \sum_{i=1}^N \log \int P(\mathbf{I}_i, Z_i; \Theta, \mathbf{k}) dZ_i, \quad (11.67)$$

where the latent vector  $Z_i$  for an observed data  $\mathbf{I}_i$  is integrated out, and  $P(\mathbf{I}_i, Z_i; \Theta, \mathbf{k})$  is the complete-data likelihood. It is well known that using squared Euclidean distance alone to train generator networks often yields blurry reconstruction results, since the precise location information of details may not be preserved, and the  $L_2$  loss in the image space leads to averaging effects among all likely locations. In order to improve the quality, we recruit an energy-based model as a critic of the generator model which serves as an actor. The energy-based model is in the form of exponential tilting of a reference distribution

$$P(\mathbf{I}; \Phi) = \frac{1}{Z(\Phi)} \exp[-f(\mathbf{I}; \Phi)] q(\mathbf{I}), \quad (11.68)$$

where  $f(\mathbf{I}; \Phi)$  is parameterized by a bottom-up ConvNet which maps an image  $\mathbf{I}$  to the feature statistics or energy,  $Z(\Phi) = \int \exp[f(\mathbf{I}; \Phi)] q(\mathbf{I}) d\mathbf{I} = \mathcal{E}_q\{\exp[f(\mathbf{I}; \Phi)]\}$  is the normalizing constant, and  $q(\mathbf{I})$  is the reference distribution such as Gaussian white noise,

$$q(\mathbf{I}) = \frac{1}{(2\pi\sigma^2)^{D/2}} \exp\left[-\frac{\|\mathbf{I}\|^2}{2\sigma^2}\right]. \quad (11.69)$$

Let  $P(\mathbf{I})$  be the underlying data distribution. Expectation with respect to  $P(\mathbf{I})$  is taken to be the average over training examples. We jointly learn the generator model and the energy-based model by introducing the following cross-entropy triangle, under a unified probabilistic framework,

$$\min_{\Theta} \max_{\Phi} T(\Theta, \Phi) = H[P(\mathbf{I}), P(\mathbf{I}; \Theta, \mathbf{k})] - H[P(\mathbf{I}), P(\mathbf{I}; \Phi)] + H[P(\mathbf{I}; \Theta, \mathbf{k}), P(\mathbf{I}; \Phi)], \quad (11.70)$$

where  $H[P, Q] = -\mathcal{E}_P[\log Q]$  denotes the cross-entropy between the two distributions.

$H[P(\mathbf{I}), P(\mathbf{I}; \Theta, \mathbf{k})]$  and  $H[P(\mathbf{I}), P(\mathbf{I}; \Phi)]$  lead to the maximum likelihood learning of the two models respectively, while  $H[\Theta, \Phi] = H[P(\mathbf{I}; \Theta, \mathbf{k}), P(\mathbf{I}; \Phi)]$  connects and modifies the learning of the two models.  $H[P(\mathbf{I}; \Theta, \mathbf{k}), P(\mathbf{I}; \Phi)]$  causes the following effect: the energy-based model criticizes the generator model by assigning lower energies to the observed examples than the synthesized examples. The generator model then improves its synthesis by lowering the energies of the synthesized examples. For details of the learning and inference please refer to [251, 252]. The deep sparse generator model is capable of joint image synthesis and reconstruction, as well as learning interpretable dictionaries at multiple layers, moreover, learns the compositional structures and primitives automatically from data. The top of Figure 11.30 show examples of reconstruction and generation for human fashion images, where the learned AND-OR tree models are shown at the bottom of Figure 11.30 in the same way as Figure 11.28). Both the reconstructed images and the generated images look sharp.

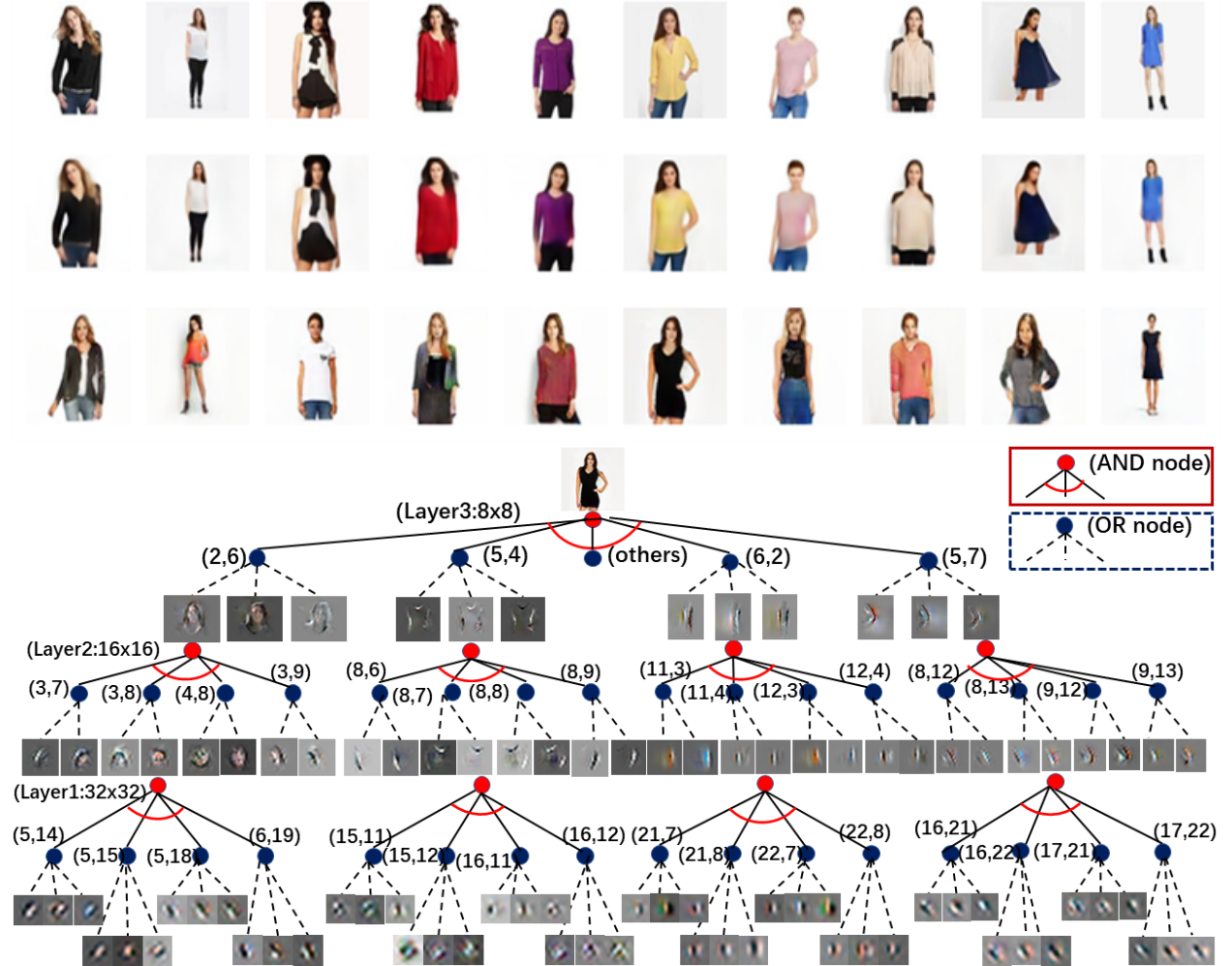


Figure 11.30: Reprinted with permission from [251]. Results on the human fashion dataset. Top: the three rows show original images, reconstructed images and generated images respectively. Bottom: the learned AND-OR tree model.

### 11.2.7 Dynamic Generator model

Let  $X = (\mathbf{I}_t, t = 1, \dots, T)$  be the observed video sequence, where  $x_t$  is a frame at time  $t$ . The dynamic generator model consists of the following two components:

$$s_t = F_\alpha(s_{t-1}, \xi_t), \quad (11.71)$$

$$x_t = G_\beta(s_t) + \epsilon_t, \quad (11.72)$$

where  $t = 1, \dots, T$ . (11.71) is the transition model, and (11.72) is the emission model.  $s_t$  is the  $d$ -dimensional hidden state vector.  $\xi_t \sim N(0, I)$  is the noise vector of a certain dimensionality. The Gaussian noise vectors  $(\xi_t, t = 1, \dots, T)$  are independent of each other. The sequence of  $(s_t, t = 1, \dots, T)$  follows a non-linear auto-regressive model, where the noise vector  $\xi_t$  encodes the randomness in the transition from  $s_{t-1}$  to  $s_t$  in the  $d$ -dimensional state space.  $F_\alpha$  is a feedforward neural network or multi-layer perceptron, where  $\alpha$  denotes the weight and bias parameters of the network. We can adopt a residual form [88] for  $F_\alpha$  to model the change of the state vector.  $x_t$  is the  $D$ -dimensional image, which is generated by the  $d$ -dimensional hidden state vector  $s_t$ .  $G_\beta$  is a top-down convolutional network (sometimes also called deconvolution network), where  $\beta$  denotes the weight and bias parameters of this top-down network.  $\epsilon_t \sim N(0, \sigma^2 I_D)$  is the residual error. We let  $\theta = (\alpha, \beta)$  denote all the model parameters.

Let  $\xi = (\xi_t, t = 1, \dots, T)$ .  $\xi$  consists of the latent random vectors that need to be inferred from  $X$ . Although  $x_t$  is generated by the state vector  $s_t$ ,  $S = (s_t, t = 1, \dots, T)$  are generated by  $\xi$ . In fact, we can write  $X = H_\theta(\xi) + \epsilon$ , where  $H_\theta$  composes  $F_\alpha$  and  $G_\beta$  over time, and  $\epsilon = (\epsilon_t, t = 1, \dots, T)$  denotes the observation errors.

Let  $p(\xi)$  be the prior distribution of  $\xi$ . Let  $p_\theta(X|\xi) \sim N(H_\theta(\xi), \sigma^2 I)$  be the conditional distribution of  $X$  given  $\xi$ , where  $I$  is the identity matrix whose dimension matches that of  $X$ . The marginal distribution is  $p_\theta(X) = \int p(\xi)p_\theta(X|\xi)d\xi$  with the latent variable  $\xi$  integrated out. We estimate the model parameter  $\theta$  by the maximum likelihood method that maximizes the observed-data log-likelihood  $\log p_\theta(X)$ , which is analytically intractable. In contrast, the complete-data log-likelihood  $\log p_\theta(\xi, X)$ , where  $p_\theta(\xi, X) = p(\xi)p_\theta(X|\xi)$ , is analytically tractable. The following identity links the gradient of the observed-data log-likelihood  $\log p_\theta(X)$  to the gradient of the complete-data log-likelihood  $\log p_\theta(\xi, X)$ :

$$\begin{aligned} \frac{\partial}{\partial \theta} \log p_\theta(X) &= \frac{1}{p_\theta(X)} \frac{\partial}{\partial \theta} p_\theta(X) \\ &= \frac{1}{p_\theta(X)} \int \left[ \frac{\partial}{\partial \theta} \log p_\theta(\xi, X) \right] p_\theta(\xi, X) d\xi \\ &= \mathcal{E}_{p_\theta(\xi|X)} \left[ \frac{\partial}{\partial \theta} \log p_\theta(\xi, X) \right], \end{aligned} \quad (11.73)$$

where  $p_\theta(\xi|X) = p_\theta(\xi, X)/p_\theta(X)$  is the posterior distribution of the latent  $\xi$  given the observed  $X$ . The above expectation can be approximated by Monte Carlo average. Specifically, we sample from the posterior distribution  $p_\theta(\xi|X)$  using the Langevin dynamics:

$$\xi^{(\tau+1)} = \xi^{(\tau)} + \frac{\delta^2}{2} \frac{\partial}{\partial \xi} \log p_\theta(\xi^{(\tau)}|X) + \delta z_\tau, \quad (11.74)$$

where  $\tau$  indexes the time step of the Langevin dynamics (not to be confused with the time step of the dynamics model,  $t$ ),  $z_\tau \sim N(0, I)$  where  $I$  is the identity matrix whose dimension matches that of  $\xi$ , and  $\xi^{(\tau)} = (\xi_t^{(\tau)}, t = 1, \dots, T)$  denotes all the sampled latent noise vectors at time step  $\tau$ .  $\delta$  is the step size of



the Langevin dynamics. We can correct for the finite step size by adding a Metropolis-Hastings acceptance-rejection step. After sampling  $\xi \sim p_\theta(\xi|X)$  using the Langevin dynamics, we can update  $\theta$  by stochastic gradient ascent

$$\Delta\theta \propto \frac{\partial}{\partial\theta} \log p_\theta(\xi, X), \quad (11.75)$$

where the stochasticity of the gradient ascent comes from the fact that we use Monte Carlo to approximate the expectation in (11.73). The learning algorithm iterates the following two steps. (1) Inference step: Given the current  $\theta$ , sample  $\xi$  from  $p_\theta(\xi|X)$  according to (12.76). (2) Learning step: Given  $\xi$ , update  $\theta$  according to (11.75). We can use a warm start scheme for sampling in step (1). Specifically, when running the Langevin dynamics, we start from the current  $\xi$ , and run a finite number of steps. Then we update  $\theta$  in step (2) using the sampled  $\xi$ . Such a stochastic gradient ascent algorithm has been analyzed by [254].

Since  $\frac{\partial}{\partial\xi} \log p_\theta(\xi|X) = \frac{\partial}{\partial\xi} \log p_\theta(\xi, X)$ , both steps (1) and (2) involve derivatives of

$$\log p_\theta(\xi, X) = -\frac{1}{2} \left[ \|\xi\|^2 + \frac{1}{\sigma^2} \|X - H_\theta(\xi)\|^2 \right] + \text{const},$$

where the constant term does not depend on  $\xi$  or  $\theta$ . Step (1) needs to compute the derivative of  $\log p_\theta(\xi, X)$  with respect to  $\xi$ . Step (2) needs to compute the derivative of  $\log p_\theta(\xi, X)$  with respect to  $\theta$ . Both can be computed by back-propagation through time. Therefore the algorithm is an alternating back-propagation through time algorithm. Step (1) can be called inferential back-propagation through time. Step (2) can be called learning back-propagation through time.

To be more specific, the complete-data log-likelihood  $\log p_\theta(\xi, X)$  can be written as (up to an additive constant, assuming  $\sigma^2 = 1$ )

$$L(\theta, \xi) = -\frac{1}{2} \sum_{t=1}^T [\|x_t - G_\beta(s_t)\|^2 + \|\xi_t\|^2]. \quad (11.76)$$

The derivative with respect to  $\beta$  is

$$\frac{\partial L}{\partial\beta} = \sum_{t=1}^T (\mathbf{I}_t - G_\beta(s_t)) \frac{\partial G_\beta(s_t)}{\partial\beta}. \quad (11.77)$$

The derivative with respect to  $\alpha$  is

$$\frac{\partial L}{\partial\alpha} = \sum_{t=1}^T (\mathbf{I}_t - G_\beta(s_t)) \frac{\partial G_\beta(s_t)}{\partial s_t} \frac{\partial s_t}{\partial\alpha}, \quad (11.78)$$

where  $\frac{\partial s_t}{\partial\alpha}$  can be computed recursively. To infer  $\xi$ , for any fixed time point  $t_0$ ,

$$\frac{\partial L}{\partial\xi_{t_0}} = \sum_{t=t_0+1}^T (\mathbf{I}_t - G_\beta(s_t)) \frac{\partial G_\beta(s_t)}{\partial s_t} \frac{\partial s_t}{\partial\xi_{t_0}} - \xi_{t_0}, \quad (11.79)$$

where  $\frac{\partial s_t}{\partial\xi_{t_0}}$  can again be computed recursively.

A minor issue is the initialization of the transition model. We may assume that  $s_0 \sim N(0, I)$ . In the inference step, we can sample  $s_0$  together with  $\xi$  using the Langevin dynamics.



It is worth mentioning the difference between our algorithm and the variational inference. While variational inference is convenient for learning a regular generator network, for the dynamic generator model, it is not a simple task to design an inference model that infers the sequence of latent vectors  $\xi = (\xi_t, t = 1, \dots, T)$  from the sequence of  $X = (\mathbf{I}_t, t = 1, \dots, T)$ . In contrast, our learning method does not require such an inference model and can be easily implemented. The inference step in our model can be done via directly sampling from the posterior distribution  $p_\theta(\xi|X)$ , which is powered by back-propagation through time. Additionally, our model directly targets maximum likelihood, while model learning via variational inference is to maximize a lower bound.

We first learn the model for dynamic textures, which are sequences of images of moving scenes that exhibit stationarity in time. We learn a separate model from each example. The video clips for training are collected from DynTex++ dataset and the Internet. Each observed video clip is prepared to be of the size 64 pixels  $\times$  64 pixels  $\times$  60 frames. The transition model is a feedforward neural network with three layers. The network takes a 100-dimensional state vector  $s_{t-1}$  and a 100-dimensional noise vector  $\xi_t$  as input and produces a 100-dimensional vector  $r_t$ , so that  $s_t = \tanh(s_{t-1} + r_t)$ . The numbers of nodes in the three layers of the feedforward neural network are  $\{20, 20, 100\}$ . The emission model is a top-down deconvolution neural network or generator model that maps the 100-dimensional state vector (i.e.,  $1 \times 1 \times 100$ ) to the image frame of size  $64 \times 64 \times 3$  by 6 layers of deconvolutions with a kernel size of 4 and up-sampling factor of 2 from top to bottom. The numbers of channels at different layers of the generator are  $\{512, 512, 256, 128, 64, 3\}$ . Batch normalization [107] and ReLU layers are added between deconvolution layers, and tanh activation function is used at the bottom layer to make the output signals fall within  $[-1, 1]$ . We use the Adam [120] for optimization with  $\beta_1 = 0.5$  and the learning rate is 0.002. We set the Langevin step size to be  $\delta = 0.03$  for all latent variables, and the standard deviation of residual error  $\sigma = 1$ . We run  $l = 15$  steps of Langevin dynamics for inference of the latent noise vectors within each learning iteration.

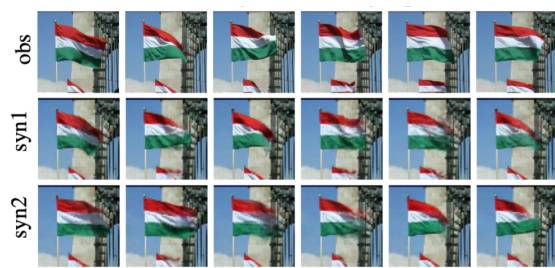
Once the model is learned, we can synthesize dynamic textures from the learned model by firstly randomly initializing the initial hidden state  $s_0$ , and then following Equation (11.71) and (11.72) to generate a sequence of images with a sequence of innovation vectors  $\{\xi_t\}$  sampled from Gaussian distribution. In practice, we use "burn-in" to throw away some iterations at the beginning of the dynamic process to ensure the transition model enters the high probability region (i.e., the state sequence  $\{s_t\}$  converges to stationarity), no matter where  $s_0$  starts from.

To speed up the training process and relieve the burden of computer memory, we can use truncated back-propagation through time in training our model. That is, we divide the whole training sequence into different non-overlapped chunks, and run forward and backward passes through chunks of the sequence instead of the whole sequence. We carry hidden states  $\{s_t\}$  forward in time forever, but only back-propagate for the length (the number of image frames) of the chunk. In this experiment, the length of the chunk is set to be 30 image frames.

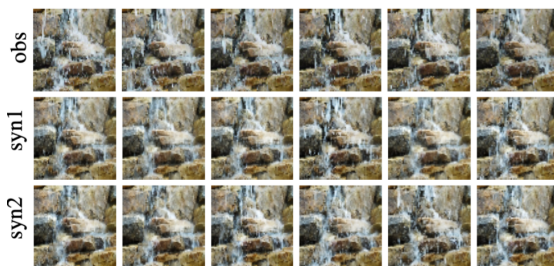
An "infinite length" dynamic texture can be synthesized from a typically "short" input sequence by just drawing "infinite" IID samples from Gaussian distribution. Figure 11.31 shows five results. For each example, the first row displays 6 frames of the observed 60-frame sequence, while the second and third rows display 6 frames of two synthesized sequences of 120 frames in length, which are generated by the learned model.



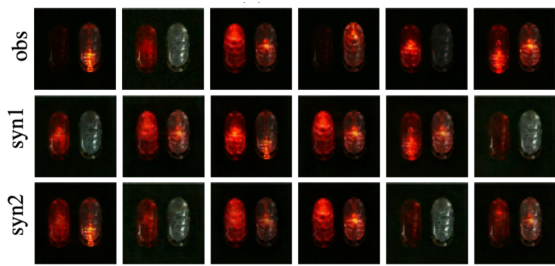
(a) burning fire heating a pot



(b) flapping flag



(c) waterfall



(d) flashing lights

Figure 11.31: Reprinted with permission from [238]. Generating dynamic textures. For each category, the first row displays 6 frames of the observed sequence, and the second and third rows show the corresponding frames of two synthesized sequences generated by the learned model.

# 12

## A Tale of Three Families: Discriminative, Generative and Descriptive Models

### 12.1 Introduction

#### 12.1.1 Three Families of Probabilistic Models

This chapter gives a general introduction to three families of probabilistic models and their connections. The first class consists of discriminative models or classifiers that are commonly used in supervised learning. The second class consists of descriptive models, also known as energy-based models, that define unnormalized probability density functions in the data space. These models are generalizations of the FRAME model introduced in the previous chapters. The third class consists of generative models that are directed top-down models that involve latent variables. The generative models are generalizations of factor analysis and its variants.

About the names of the models, we use the term “generative models” in a much narrower sense than in the current literature. They refer to top-down models that consist of latent variables, so that the examples can be directly generated. As to the “descriptive models”, they refer to the energy-based models or deep FRAME model introduced in the previous chapter. They only describe the examples but they cannot directly generate the examples. The generative task is left to iterative MCMC sampling algorithms. Therefore these models are not literally generative, and that is why we call them descriptive.

In the previous chapter, we introduced the descriptive models and generative models for image and video data, and the associated maximum likelihood learning algorithms. This chapter will give a more general and advanced treatment. We shall still emphasize the maximum likelihood learning algorithm, but we shall focus on a very efficient variation powered by short-run MCMC sampling for inference and synthesis. Meanwhile, we shall also present various joint training schemes, such as variational learning and adversarial learning. We shall make this chapter self-contained, so that readers who are interested in the development in the deep learning era can read this chapter in isolation.

**Notation.** We shall adopt the notation commonly used in the current literature. We use  $x$  to denote the training example, e.g., an image or a sentence. We use  $z$  to denote the latent variables in the generative model. We use  $y$  to denote the output in the discriminative model, e.g., image category. We use  $\theta$  to denote the model parameters. We use the notation  $\nabla_x$  and  $\nabla_\theta$  to denote  $\frac{\partial}{\partial x}$  and  $\frac{\partial}{\partial \theta}$  respectively. For a function  $h(\theta)$ , its derivative at a fixed value, say,  $\theta_t$ , is denoted  $\nabla_\theta h(\theta_t)$ . We use KL to denote the Kullback-Leibler divergence.

### 12.1.2 Supervised, Unsupervised, and Semi-supervised Learning

Supervised learning refers to the situation where both the input  $x$  and the output  $y$  are given, and we want learn to predict  $y$  based on  $x$ . More formally, we learn a discriminative model  $p(y|x)$  by maximum likelihood, i.e., we maximize the average of  $\log p(y|x)$  over the model parameters where the average is over the training set  $\{x, y\}$ . The limitation of supervised learning is that it can be expensive and time consuming to obtain  $y$  in the form of label or annotation.

Unsupervised learning refers to the situation where only the input  $x$  is given, but the output  $y$  is unavailable. In that case, we can learn a descriptive model or a generative model, again by maximum likelihood, but we maximize the average of  $\log p(x)$  over the model parameters, where the average is over the training set  $\{x\}$ , instead of the average of  $\log p(y|x)$ , as  $y$  is not available. The descriptive model specifies  $p(x)$  up to an unknown normalizing constant, and it is closely related to the discriminative model through the Bayes rule. For the generative model,  $p(x)$  is implicit because it involves integrating out the latent variables  $z$ . The latent  $z$  can be inferred from the input  $x$ .

Semi-supervised learning refers to the situation between supervised and unsupervised learning, where there are a small number of labeled examples where both  $x$  and  $y$  are given, and there are a large number of unlabeled examples where only  $x$  is given. In that case, we can again learn the model by maximum likelihood, where we maximize the sum of  $\log p(y|x)$  over the labeled examples and  $\log p(x)$  over the unlabeled examples. Thus probabilistic modeling provides a unified likelihood-based framework for supervised, unsupervised and semi-supervised learning.

There is also self-supervised learning, which is to translate unsupervised learning into supervised learning. Specifically, even if we are only given  $x$  without  $y$ , we can nonetheless create a task where we artificially introduce  $y$  for a modification of  $x$  that depends on  $y$ , and we then learn  $p(y|x)$  instead of  $p(x)$ . This type of learning can be more formally treated as learning descriptive model by various conditional likelihood.

### 12.1.3 Short-run MCMC for Synthesis and Inference

Although likelihood-based learning with probabilistic models is a principled framework for supervised, unsupervised and semi-supervised learning, the bottleneck for likelihood-based learning for unsupervised learning is that the derivative of the log-likelihood function  $\log p(x)$  usually involves intractable integrals or expectations, which require expensive MCMC sampling. A lot of effort has been spent on getting around this obstacle.

We advocate short-run MCMC, i.e., running MCMC such as Langevin dynamics from a fixed initial noise distribution for a fixed number of steps, for inference and synthesis computations. This is easily affordable on modern computing platforms. It can also be justified as a modification or perturbation of the maximum likelihood learning.

Short-run MCMC is particularly appealing for learning models with multiple layers of latent variables organized in complex architectures, because top-down feedbacks and lateral inhibitions between the latent variables at different layers can automatically emerge in short-run MCMC. The short-run Langevin dynamics can also be compared with attractor dynamics that is a commonly assumed framework for modeling neural computations. In computational neuroscience, people usually start from postulating attractor dynamics, but in our framework, such computational dynamics is derived as inference or synthesis computation guided by a probabilistic model.

One can also run persistent Markov chains, i.e., in each learning iteration, we initialize finite step MCMC from the samples generated in the previous learning iteration. We can manage to make the learning algorithm work. But it has the following problems: (1) Theoretically, the convergence of the learning algorithm with persistent chains is hard to analyze or justify, and may require unverifiable or unrealistic assumptions about

the Markov transition kernel. (2) Practically, the algorithm is awkward for mini-batch training, which is necessary for learning from big data. In mini-batch training, the values of the parameters keep changing over the mini-batches within the same learning epoch, so that in the same learning epoch, different mini-batches initialize the chains from samples generated with different values of the parameters. Moreover, after learning, in the testing stage, we cannot run the same persistent chains, thus the learning and testing stages use different sampling schedules. In contrast, the noise initialized short-run MCMC does not have such problems.

#### 12.1.4 Deep Neural Networks as Function Approximators

All three classes of models can be parametrized by deep neural networks [124, 127], which are compositions of multiple layers of linear transformations and coordinate-wise nonlinear link functions.

Specifically, consider a nonlinear transformation  $f(x)$  that can be decomposed recursively as  $s_l = W_l h_{l-1} + b_l$ , and  $h_l = r_l(s_l)$ , for  $l = 1, \dots, L$ , with  $f(x) = h_L$  and  $h_0 = x$ .  $W_l$  is a weight matrix at layer  $l$ , and  $b_l$  is the bias vector at layer  $l$ . Both  $s_l$  and  $h_l$  are vectors of the same dimensionality, and  $r_l$  is a one-dimensional nonlinear link function, or rectification function, that is applied coordinate-wise.  $f(x)$  is a recursive composition of generalized linear model (GLM) structures.

Modern deep networks usually use  $r_l(s) = \max(0, s)$ , the rectified linear unit or ReLU. For such nonlinear link functions,  $f(x)$  is a multivariate linear spline where the linear pieces are recursively partitioned. This is similar to but more general than the recursive partitions in classification and regression trees (CART) [20] and multivariate adaptive regression splines (MARS) [61].

There are two special classes of neural networks. One consists of convolutional neural networks [124, 127], which are commonly applied to images, where the same linear transformations are applied around each pixel locally. The other class consists of recurrent neural networks [94], which are commonly applied to sequence data such as speech and natural language. Recently the Transformer self-attention architecture has gained popularity in natural language processing as well as computer vision.

Deep neural networks are powerful function approximators that can approximate highly non-linear high-dimensional functions by interpolating training examples. Modern deep networks are highly over-parametrized, meaning that the number of parameters greatly exceeds the number of training examples. Thus they have enough capacity to fit the training data, yet they tend not to overfit the training data because the networks are learned by stochastic gradient descent algorithm where the gradient is computed via back-propagation. The stochastic gradient descent algorithm provides implicit regularization [10, 204].

#### 12.1.5 Learned Computation

Because of the strong approximation capacity, the boundary between representation and computation is rather blurred, because a deep network can approximate an iterative algorithm. Sometimes this is called learned computation.

In fact, the residual network can be considered a finite step iterative algorithm. It is of the form  $x_{l+1} = x_l + f_l(x_l)$ , where  $l$  indexes the layer. Meanwhile,  $l$  may also be interpreted as time step of an iterative algorithm, i.e., we can also write  $x_{t+1} = x_t + f_t(x_t)$ , which is to model iterative updating. In general, it can be interpreted as a mixture of both, i.e., there is actually a small number of layers, and each layer is computed by a finite step iterative algorithm.

The transformer model can also be considered a finite step iterative algorithm that iteratively updates the vector representations of the words of an input sentence through a self-attention mechanism where the words pay attention to and gather information from each other. The graph convolutional network can learn the iterative message passing mechanism where the nodes of the graph send messages to each other.

In the above iterative updating mechanisms, there is no need to know the objective functions of these iterative mechanisms. They can be embedded into a classifier and be trained by the classification loss via back-propagation through time.

### 12.1.6 Amortized Computation for Synthesis and Inference Sampling

Even if there is an objective function, we can still learn a deep network that directly maps the input to an approximate solution. Sometimes this is called amortized network, which seeks to approximate an iterative algorithm of multiple time steps.

In the case of generative model, recall that we can use short-run MCMC as approximate sampler for synthesis and inference. We can also learn a network that produces the samples directly. In the case of posterior sampling, this is referred to as variational inference model. In fact, the short-run MCMC can be considered a noise injected residual network.

When there are multiple layers of latent variables, designing a network for approximate inference sampling can be a non-trivial task, whereas short-run MCMC remains automatic.

### 12.1.7 Distributed and Symbolic Representations

Deep neural networks are based on continuous vectors and weight matrices. They are highly interpolative and amenable to gradient-based computations. On the other hand, high level reasoning can also be highly symbolic, with symbols, logic and grammars. For a dictionary of symbols, each symbol can be represented by a one-hot binary vector, and a small subset of symbols selected from the dictionary can be represented by a sparse binary vector. This is in contrast to the vectors in deep networks, which are continuous and dense. Such vectors are called distributed representations. Sometimes they are also referred to as embeddings. For instance, the word2vec model represents each word by a dense vector, and this means we embed the words in a continuous Euclidean space.

It is still unclear how to unify symbolic and dense representations. Sometimes this is referred to the contrast between symbolism and connectionism. It is likely that there is a duality or complementarity between sparse vectors and dense vectors, and each is more convenient and efficient than the other depending on the scenario.

### 12.1.8 Perturbations of KL-divergence: a Unifying Theoretical Framework

A unifying theoretical device for studying various learning methods is to perturb the Kullback-Leibler divergence for maximum likelihood by other Kullback-Leibler divergences. This scheme consists of three Kullback-Leibler divergences. (1) KL-divergence underlying maximum likelihood learning. This is the target of the perturbations. (2) KL-divergence underlying synthesis sampling. (3) KL-divergence underlying inference sampling. (2) and (3) are perturbations that are applied to (1). The sign in front of (2) is negative, and the sign in front of (3) is positive.

The above theoretical framework explains the following learning algorithms. (1) The maximum likelihood learning algorithm. (2) The learning algorithm based on short-run MCMC for synthesis and inference. (3) The learning methods based on learned networks for synthesis and inference, including adversarial learning and variational learning.

### 12.1.9 Kullback-Leibler in Two Directions

To be more specific, recall that for two probability densities  $p(x)$  and  $q(x)$ , we define

$$\text{KL}(p\|q) = \mathcal{E}_p \left[ \log \frac{p(x)}{q(x)} \right] = \int p(x) \log \frac{p(x)}{q(x)} dx. \quad (12.1)$$

The KL-divergence appears in two scenarios.

(1) **Maximum likelihood learning.** Suppose the training examples  $x_i \sim p_{\text{data}}(x)$  are independent for  $i = 1, \dots, n$ . Suppose we want to learn a model  $p_\theta(x)$ . The log-likelihood function is

$$L(\theta) = \frac{1}{n} \sum_{i=1}^n \log p_\theta(x_i) \rightarrow \mathcal{E}_{p_{\text{data}}} [\log p_\theta(x)]. \quad (12.2)$$

Thus for big  $n$ , maximizing  $L(\theta)$  is equivalent to minimizing

$$\text{KL}(p_{\text{data}}\|p_\theta) = -\text{entropy}(p_{\text{data}}) - \mathcal{E}_{p_{\text{data}}} [\log p_\theta(x)] \doteq -\text{entropy}(p_{\text{data}}) - L(\theta), \quad (12.3)$$

where  $\mathcal{E}_{p_{\text{data}}}$  can be approximated by averaging over  $\{x_i\}$ . We can think of it as projecting  $p_{\text{data}}$  onto the model space  $\{p_\theta, \forall \theta\}$ .

For the rest of this chapter, for notational simplicity, we will not distinguish between  $\mathcal{E}_{p_{\text{data}}}$  and sample average over  $\{x_i\}$ , and we will treat  $\text{KL}(p_{\text{data}}\|p_\theta)$  as the loss function for maximum likelihood learning.

(2) **Variational approximation.** Suppose we have a target distribution  $p_{\text{target}}$ , and we know  $p_{\text{target}}$  up to a normalizing constant, e.g.,  $p_{\text{target}}(x) = \exp(f(x))/Z$ , where we know  $f(x)$  but the normalizing constant  $Z = \int \exp(f(x))dx$  is analytically intractable. Suppose we want to approximate it by a distribution  $q_\phi$ . We can find  $\phi$  by minimizing

$$\text{KL}(q_\phi\|p_{\text{target}}) = \mathcal{E}_{q_\phi} [\log q_\phi(x)] - \mathcal{E}_{q_\phi} [f(x)] + \log Z. \quad (12.4)$$

This time, we place  $q_\phi$  on the left hand side, and  $p_{\text{target}}$  on the right hand side of the KL-divergence, because  $p_{\text{target}}$  is accessible only through  $f(x)$ . The above minimization does not require knowledge of  $\log Z$ .

The behaviors of  $\min_\theta \text{KL}(p_{\text{data}}\|p_\theta)$  in (1) and  $\min_\phi \text{KL}(q_\phi\|p_{\text{target}})$  in (2) are different. In (1),  $p_\theta$  tends to cover all the modes of  $p_{\text{data}}$ , because  $\text{KL}(p_{\text{data}}\|p_\theta)$  is expectation with respect to  $p_{\text{data}}$ . In (2),  $q_\phi$  tends to focus on some major modes of  $p_{\text{target}}$ , while ignoring the minor modes, because  $\text{KL}(q_\phi\|p_{\text{target}})$  is expectation with respect to  $q_\phi$ .

In the perturbation scheme mentioned in the previous subsection, the KL-divergence for maximum likelihood is 12.3. The perturbations are of the form in 12.4.

## 12.2 Descriptive Model

### 12.2.1 Model and Origin

Let  $x$  be a training example, e.g., an image or a sentence. A descriptive model specifies an unnormalized probability density function

$$p_\theta(x) = \frac{1}{Z_\theta} \exp(f_\theta(x)), \quad (12.5)$$

where  $f_\theta(x)$  is parametrized by a deep network, with  $\theta$  collecting all the weight and bias parameters.  $Z_\theta = \int \exp(f_\theta(x))dx$  is the normalizing constant.

Such a model originated from statistical mechanics, and is called the Gibbs distribution, where  $x$  is the state or configuration of a physical system, and  $-f_\theta(x)$  is the energy function of the state, so that the lower energy states are more likely to be observed. For that reason, the above model is also called energy-based model in the literature [31, 66, 90, 146, 165, 166, 168, 244, 247, 248].

In classical mechanics, the configuration  $x(t)$  evolves deterministically over time  $t$  according to a partial differential equation. Then where does the probability distribution come from? We may consider the ensemble or population  $(x(t), t \in [t_0, t_1])$ , for a big enough burn-in time  $t_0$  and long enough duration  $t_1 - t_0$ . For a random time  $t \sim \text{Uniform}[t_0, t_1]$ ,  $x(t)$  follows a probability distribution  $p(x)$ , and it can be modeled by a Gibbs distribution.

The quantity  $Z_\theta$  is called partition function in statistical mechanics. An important identity is

$$\nabla_\theta \log Z_\theta = \mathcal{E}_{p_\theta}[\nabla_\theta f_\theta(x)]. \quad (12.6)$$

The non-differentiability of  $\log Z_\theta$  underlies the phase transition phenomena in statistical physics.

### 12.2.2 Gradient-based Sampling

For high dimensional  $x$ , such as image, sampling from  $p_\theta(x)$  requires MCMC, such as Langevin dynamics or Hamiltonian Monte Carlo. The Langevin dynamics iterates

$$x_{t+1} = x_t + s \nabla_x f_\theta(x_t) + \sqrt{2s} e_t, \quad (12.7)$$

where  $s$  is the step size, and  $e_t \sim \mathcal{N}(0, I)$  is the Gaussian white noise term. The Langevin dynamics has a gradient ascent term  $\nabla_x f_\theta(x_t)$ , and  $e_t$  is the diffusion term for randomness. As  $s \rightarrow 0$  and  $t \rightarrow \infty$ , the distribution of  $x_t$  converges to  $p_\theta(x)$ .

We can write the Langevin dynamics in continuous time as

$$x_{t+\Delta t} = x_t + \frac{1}{2} \nabla_x f_\theta(x_t) \Delta t + \sqrt{\Delta t} e_t, \quad (12.8)$$

or more formally,

$$dx_t = \frac{1}{2} \nabla_x f_\theta(x_t) dt + dB_t, \quad (12.9)$$

where  $dB_t$  plays the role of  $\sqrt{\Delta t} e_t$ .

Let  $p_t$  be the distribution of  $x_t$ . Then according to the Fokker-Planck equation, we have

$$\nabla_t p_t(x) = \frac{1}{2} [\nabla_x (f_\theta(x) p_t(x)) + \nabla_x^2 p_t(x)]. \quad (12.10)$$

$p_\theta(x)$  is the solution to  $\nabla_t p_t(x) = 0$ , i.e., the stationary distribution. In terms of variational approximation,

$$\text{KL}(p_t \| p_\theta) = -\text{entropy}(p_t) - \mathcal{E}_{p_t}[f_\theta(x)] + \log Z_\theta \rightarrow 0 \quad (12.11)$$

monotonically at  $t \rightarrow \infty$ . The gradient term in the Langevin dynamics increases  $f_\theta(x)$  or decreases energy, while the noise term  $e_t$  increases the entropy of  $p_t$ .



### 12.2.3 Expressive Power

The descriptive model has strong expressive power, because it only needs to specify a scalar-valued function  $f_\theta(x)$ , to describe what it wants, without bothering with how to get what it wants.  $f_\theta(x)$  is like an objective function (or value function, or constraints, or rules). The descriptive model is only responsible for specifying the objective function, and is not responsible for optimizing the objective function or providing near-optimal solutions. The latter task is left to MCMC sampling. As a result, a simple descriptive model  $p_\theta(x)$  or the objective function  $f_\theta(x)$  can explain rich patterns and complex behaviors.

The descriptive model has been used for inverse reinforcement learning, where  $-f_\theta(x)$  serves as the cost function. It has also been used for Markov logic network, where  $f_\theta(x)$  combines logical rules.

Thus a scalar-valued energy function has strong explaining power.

### 12.2.4 Maximum Likelihood Estimation (MLE) and Analysis by Synthesis

The descriptive model  $p_\theta(x)$  can be learned by maximum likelihood estimation (MLE). The log-likelihood is the average of

$$\log p_\theta(x) = f_\theta(x) - \log Z_\theta, \quad (12.12)$$

where the average is over the training set  $\{x\}$ . The gradient of  $\log p_\theta(x)$  with respect to  $\theta$  is

$$\delta_\theta(x) = \nabla_\theta \log p_\theta(x) = \nabla_\theta f_\theta(x) - \mathcal{E}_{p_\theta(x)}[\nabla_\theta f_\theta(x)], \quad (12.13)$$

where

$$\nabla_\theta \log Z_\theta = \mathcal{E}_{p_\theta(x)}[\nabla_\theta f_\theta(x)]. \quad (12.14)$$

Suppose we observe training examples  $\{x_i, i = 1, \dots, n\} \sim p_{\text{data}}$ , where  $p_{\text{data}}$  is the data distribution. For large  $n$ , the sample average over  $\{x_i\}$  approximates the expectation with respect to  $p_{\text{data}}$ . For notational convenience, we treat the sample average and the expectation as the same.

The log-likelihood is

$$L(\theta) = \frac{1}{n} \sum_{i=1}^n \log p_\theta(x_i) \doteq \mathcal{E}_{p_{\text{data}}}[\log p_\theta(x)]. \quad (12.15)$$

The derivative of the log-likelihood is

$$L'(\theta) = \mathcal{E}_{p_{\text{data}}}[\delta_\theta(x)] = \mathcal{E}_{p_{\text{data}}}[\nabla_\theta f_\theta(x)] - \mathcal{E}_{p_\theta}[\nabla_\theta f_\theta(x)] \quad (12.16)$$

$$\doteq \frac{1}{n} \sum_{i=1}^n \nabla_\theta f_\theta(x_i) - \frac{1}{n} \sum_{i=1}^n \nabla_\theta f_\theta(x_i^-), \quad (12.17)$$

where  $x_i^- \sim p_\theta(x)$  for  $i = 1, \dots, n$  are the generated examples from the current model  $p_\theta(x)$ .

The above equation leads to the “analysis by synthesis” learning algorithm. At iteration  $t$ , let  $\theta_t$  be the current model parameters. We generate synthesized examples  $x_i^- \sim p_{\theta_t}(x)$  for  $i = 1, \dots, n$ . Then we update  $\theta_{t+1} = \theta_t + \eta_t L'(\theta_t)$ , where  $\eta_t$  is the learning rate, and  $L'(\theta_t)$  is the statistical difference between the synthesized examples and observed examples. [244] is perhaps the first to study the MLE of energy-based model parametrized by modern convolutional neural network.

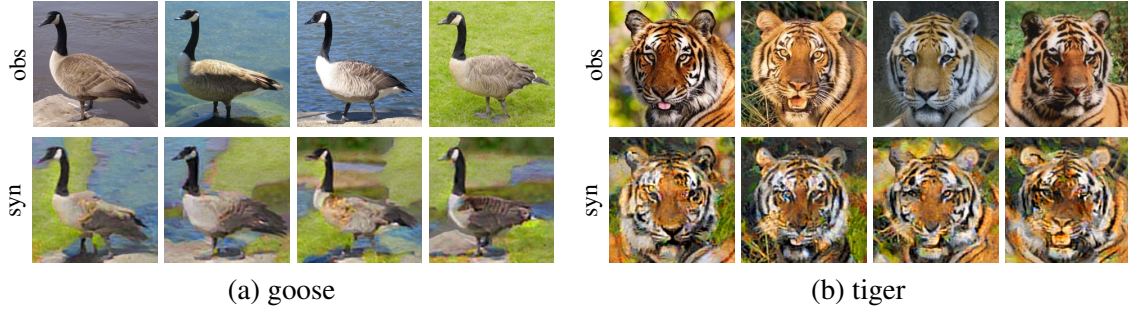


Figure 12.1: Reprinted with permission from [244]. Learning the descriptive model by maximum likelihood: (a) goose (b) tiger. For each category, the first row displays four of the training images, and the second row displays four of the images generated by the learning algorithm.  $f_\alpha(x)$  is parameterized by a four-layer bottom-up deep network, where the first layer has  $100\ 7 \times 7$  filters with sub-sampling size 2, the second layer has  $64\ 5 \times 5$  filters with subsampling size 1, the third layer has  $20\ 3 \times 3$  filters with sub-sampling size 1, and the fourth layer is a fully connected layer with a single filter that covers the whole image. The number of parallel chains for Langevin sampling is 16, and the number of Langevin iterations between every two consecutive updates of parameters is 10. The training images are  $224 \times 224$  pixels.

### 12.2.5 Objective Function and Estimating Equation

The maximum likelihood learning minimizes the Kullback-Leibler divergence  $\text{KL}(p_{\text{data}} \| p_\theta)$  over  $\theta$ . Geometrically, it is to project  $p_{\text{data}}$  onto the manifold formed by  $\{p_\theta, \forall \theta\}$ .

The maximum likelihood learning algorithm converges to the solution to the following estimating equation:

$$\mathcal{E}_{p_\theta} [\nabla_\theta f_\theta(x)] = \mathcal{E}_{p_{\text{data}}} [\nabla_\theta f_\theta(x)], \quad (12.18)$$

where the model matches the data in terms of the expectation of  $\nabla_\theta f_\theta(x)$ .

For the FRAME model or in general the exponential family model,  $f_\theta(x) = \langle \theta, h(x) \rangle$  for feature vector  $h(x)$ , hence  $\nabla_\theta f_\theta(x) = h(x)$ , and  $L'(\theta) = \mathcal{E}_{p_{\text{data}}} [h(x)] - \mathcal{E}_{p_\theta} [h(x)]$ . The maximum likelihood estimating equation is  $\mathcal{E}_{p_\theta} [h(x)] = \mathcal{E}_{p_{\text{data}}} [h(x)]$ , i.e., matching feature statistics. For general  $f_\theta(x)$ , we may still consider  $\nabla_\theta f_\theta(x)$  as a feature vector. It is related to the neural tangent kernel in deep learning theory.

### 12.2.6 Perturbation of KL-divergence

Define  $D(\theta) = \text{KL}(p_{\text{data}} \| p_\theta)$ . It is the loss function of MLE. To understand the MLE learning algorithm, let  $\theta_t$  be the estimate at iteration  $t$ . Let us consider the following perturbation of  $D(\theta)$ ,

$$S(\theta) = D(\theta) - \text{KL}(p_{\theta_t} \| p_\theta) = \text{KL}(p_{\text{data}} \| p_\theta) - \text{KL}(p_{\theta_t} \| p_\theta). \quad (12.19)$$

$S(\theta)$  is the surrogate objective function for  $D(\theta)$  at iteration  $t$ . It is simpler than  $D(\theta)$ , because the  $\log Z_\theta$  term gets canceled, and the gradient can be more easily computed.

The perturbation term  $\text{KL}(p_{\theta_t} \| p_\theta)$ , as a function of  $\theta$ , with  $\theta_t$  fixed, has the following properties. (1) It achieves minimum zero at  $\theta = \theta_t$ . (2) Its derivative is zero at  $\theta = \theta_t$ . As a result,  $S(\theta_t) = D(\theta_t)$ , and  $S'(\theta_t) = D'(\theta_t)$ . Geometrically,  $S(\theta)$  and  $D(\theta)$  touch each other at  $\theta_t$  and they are co-tangent at  $\theta_t$ . Since

$$S(\theta) = -\mathcal{E}_{p_{\text{data}}} [f_\theta(x)] + \mathcal{E}_{p_{\theta_t}} [f_\theta(x)] - \text{entropy}(p_{\text{data}}) + \text{entropy}(p_{\theta_t}), \quad (12.20)$$

where  $\log Z_\theta$  gets canceled, we have

$$-S'(\theta) = \mathcal{E}_{p_{\text{data}}} [\nabla_\theta f_\theta(x)] - \mathcal{E}_{p_{\theta_t}} [\nabla_\theta f_\theta(x)]. \quad (12.21)$$

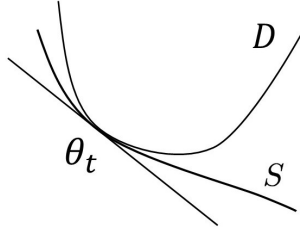


Figure 12.2: Reprinted with permission from [86]. The surrogate  $S$  minorizes (lower bounds)  $D$ , and they touch each other at  $\theta_t$  with the same tangent.

Thus

$$-D'(\theta_t) = -S'(\theta_t) = \mathcal{E}_{p_{\text{data}}}[\delta_{\theta_t}(x)] = \mathcal{E}_{p_{\text{data}}}[\nabla_{\theta} f_{\theta_t}(x)] - \mathcal{E}_{p_{\theta_t}}[\nabla_{\theta} f_{\theta_t}(x)]. \quad (12.22)$$

This justifies the MLE learning algorithm.

We shall use this perturbation scheme repeatedly, where we perturb the MLE loss function  $D(\theta) = \text{KL}(p_{\text{data}} \| p_{\theta})$  to a simpler surrogate objective function  $S(\theta)$  by subtracting or adding other KL-divergence terms. This enables us to theoretically understand other learning methods that are modifications of MLE learning.

### 12.2.7 Self-adversarial Interpretation

$S(\theta) = \text{KL}(p_{\text{data}} \| p_{\theta}) - \text{KL}(p_{\theta_t} \| p_{\theta})$  leads to an adversarial interpretation. When we update  $\theta$  by following the gradient of  $S(\theta)$  at  $\theta = \theta_t$ , we want  $p_{\theta}$  to move away from  $p_{\theta_t}$  and move towards  $p_{\text{data}}$ . That is, the model  $p_{\theta}$  criticizes its current version  $p_{\theta_t}$  by comparing  $p_{\theta_t}$  to  $p_{\text{data}}$ . The model serves as both generator and discriminator if we compare it to GAN (generative adversarial networks). In contrast to GAN, the learning algorithm is MLE, which in general does not suffer from issues such as mode collapsing and instability, as it does not involve the competition between two separate networks.

### 12.2.8 Short-run MCMC for Synthesis

$p_{\theta}(x)$  can be highly multi-modal, and gradient-based MCMC may get trapped in local modes, so that different Markov chains initialized from different starting points do not mix. We now consider the learning algorithm based on short-run MCMC [168].

The short-run MCMC is

$$x_0 \sim p_0(x), \quad x_{k+1} = x_k + s \nabla_x f_{\theta}(x_k) + \sqrt{2se_k}, \quad k = 1, \dots, K, \quad (12.23)$$

where we initialize the Langevin dynamics from a fixed diffused noise distribution  $p_0(x)$ , and we run a fixed number of  $K$  steps. Let  $\tilde{p}_{\theta}(x)$  be the distribution of  $x_K$ . We use  $x_K$  as the synthesized example for approximate maximum likelihood learning.

For each  $x$ , we define

$$\tilde{\delta}_{\theta}(x) = \nabla_{\theta} f_{\theta}(x) - \mathcal{E}_{\tilde{p}_{\theta}(x)}[\nabla_{\theta} f_{\theta}(x)], \quad (12.24)$$

and modify the learning algorithm to

$$\theta_{t+1} = \theta_t + \eta_t \mathcal{E}_{p_{\text{data}}}[\tilde{\delta}_{\theta_t}(x)] = \theta_t + \eta_t (\mathcal{E}_{p_{\text{data}}}[\nabla_{\theta} f_{\theta}(x)] - \mathcal{E}_{\tilde{p}_{\theta}}[\nabla_{\theta} f_{\theta}(x)]). \quad (12.25)$$



Figure 12.3: Reprinted with permission from [168]. Synthesis by short-run MCMC: Generating synthesized examples by running 100 steps of Langevin dynamics initialized from uniform noise for CelebA ( $64 \times 64$ ).

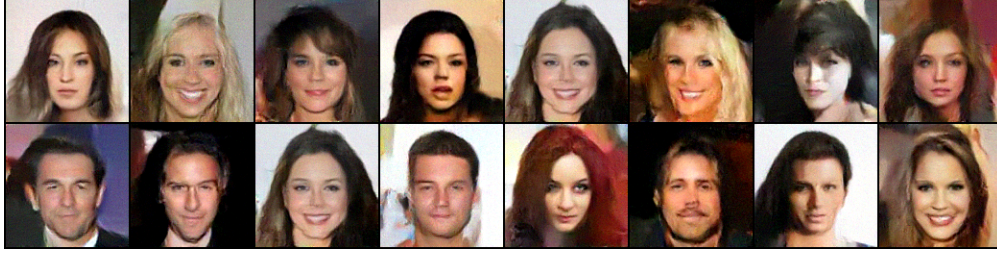


Figure 12.4: Reprinted with permission from [168]. Synthesis by short-run MCMC: Generating synthesized examples by running 100 steps of Langevin dynamics initialized from uniform noise for CelebA ( $128 \times 128$ ).

### 12.2.9 Objective Function and Estimating Equation

The following are justifications for the learning algorithm based on short-run MCMC synthesis.

(1) **Objective function.** Again we use perturbation of KL-divergence. At iteration  $t$ , with  $\theta_t$  fixed, the learning algorithm follows the gradient of the following perturbation of  $D(\theta)$  at  $\theta = \theta_t$ :

$$S(\theta) = D(\theta) - \text{KL}(\tilde{p}_{\theta_t} \| p_{\theta}) = \text{KL}(p_{\text{data}} \| p_{\theta}) - \text{KL}(\tilde{p}_{\theta_t} \| p_{\theta}), \quad (12.26)$$

so that  $\theta_{t+1} = \theta_t + \eta_t S'(\theta_t)$ , where  $\eta_t$  is the step size, and

$$-S'(\theta) = \mathcal{E}_{p_{\text{data}}}[\nabla_{\theta} f_{\theta}(x)] - \mathcal{E}_{\tilde{p}_{\theta_t}}[\nabla_{\theta} f_{\theta}(x)]. \quad (12.27)$$

$$-S'(\theta_t) = \mathcal{E}_{p_{\text{data}}}[\tilde{\delta}_{\theta_t}(x)] = \mathcal{E}_{p_{\text{data}}}[\nabla_{\theta} f_{\theta_t}(x)] - \mathcal{E}_{\tilde{p}_{\theta_t}}[\nabla_{\theta} f_{\theta_t}(x)]. \quad (12.28)$$

Compared to the perturbation of KL-divergence in MLE learning, we use  $\tilde{p}_{\theta_t}$  instead of  $p_{\theta_t}$ . While sampling  $p_{\theta_t}$  can be impractical if it is multi-modal, sampling  $\tilde{p}_{\theta_t}$  is practical and exact because it is a short-run MCMC.

Note that  $S'(\theta_t) \neq D'(\theta_t)$ , because  $\tilde{p}_{\theta_t} \neq p_{\theta_t}$ . Thus the learning gradient based on short-run MCMC is biased from that of MLE. As a result, the learned  $p_{\theta}$  based on short-run MCMC may be biased from MLE.

$S(\theta)$  indicates that we need to minimize  $\text{KL}(\tilde{p}_{\theta} \| p_{\theta})$  in order to minimize the bias relative to the maximum likelihood learning. We can do that by increasing  $K$ , because  $\text{KL}(\tilde{p}_{\theta} \| p_{\theta})$  decreases monotonically to zero as  $K$  increases. With fixed  $K$ , we can also optimize the step size  $s$  of the Langevin dynamics.

(2) **Estimating equation.** The learning algorithm converges to the solution to the following estimating equation:

$$\mathcal{E}_{\tilde{p}_{\theta}}[\nabla_{\theta} f_{\theta}(x)] = \mathcal{E}_{p_{\text{data}}}[\nabla_{\theta} f_{\theta}(x)], \quad (12.29)$$

which is a perturbation of the maximum likelihood estimating equation where we replace  $p_{\theta}$  by  $\tilde{p}_{\theta}$ .

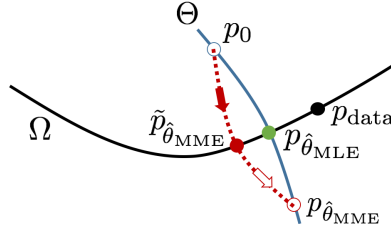


Figure 12.5: Reprinted with permission from [168]. The blue curve illustrates the model distributions corresponding to different values of parameter  $\theta$ . The black curve illustrates all the distributions that match  $p_{\text{data}}$  (black dot) in terms of  $\mathcal{E}[h(x)]$ . The MLE  $p_{\hat{\theta}_{\text{MLE}}}$  (green dot) is the intersection between  $\Theta$  (blue curve) and  $\Omega$  (black curve). The MCMC (red dotted line) starts from  $p_0$  (hollow blue dot) and runs toward  $p_{\hat{\theta}_{\text{MLE}}}$  (hollow red dot), but the MCMC stops after  $K$ -step, reaching  $\tilde{p}_{\hat{\theta}_{\text{MME}}}$  (red dot), which is the learned short-run MCMC.

Thus even if the learned  $p_\theta$  may be biased from MLE, the resulting short-run MCMC  $\tilde{p}_\theta$  can nonetheless be considered a valid model, in that it matches  $p_{\text{data}}$  in terms of expectations of  $\nabla_\theta f_\theta(x)$ . Recall in the case of FRAME model where  $f_\theta(x) = \langle \theta, h(x) \rangle$ ,  $\nabla_\theta f_\theta(x) = h(x)$ , i.e., the learned short-run MCMC  $\tilde{p}_\theta$  matches  $p_{\text{data}}$  in terms of expectations of  $h(x)$ . In general,  $\nabla_\theta f_\theta(x)$  may be considered a generalized version of feature vector  $h(x)$ . It underlies the neural tangent kernel which is an important object in deep learning theory. Thus we may justify the learned short-run MCMC  $\tilde{p}_\theta$  as a generalized moment matching estimator  $\hat{\theta}_{\text{MME}}$ .

In fact, the generalized moment matching explains the synthesis ability of the descriptive model and various learning schemes in general.



Figure 12.6: Reprinted with permission from [168]. Interpolation by short-run MCMC resembling a generator or flow model: The transition depicts the sequence  $F(z_\rho)$  with interpolated noise  $z_\rho = \rho z_1 + \sqrt{1 - \rho^2} z_2$  where  $\rho \in [0, 1]$  on CelebA ( $64 \times 64$ ). Left:  $F(z_1)$ . Right:  $F(z_2)$ .

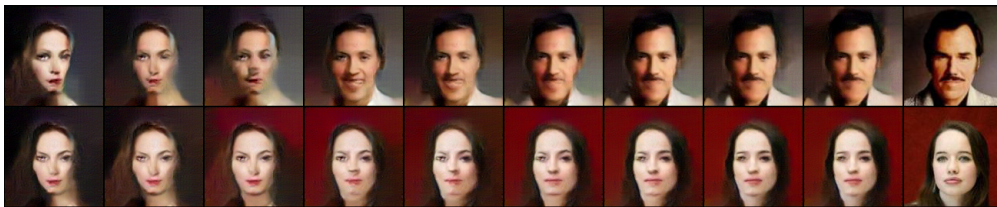


Figure 12.7: Reprinted with permission from [168]. Reconstruction by short-run MCMC resembling a generator or flow model: The transition depicts  $F(z_t)$  over time  $t$  from random initialization  $t = 0$  to reconstruction  $t = 200$  on CelebA ( $64 \times 64$ ). Left: Random initialization. Right: Observed examples.

The short-run Langevin dynamics can be considered a noise-injected RNN or noise-injected residual network. It can also be considered a flow-based model. Specifically, we can write  $x_K = F(x_0, e)$ , where  $e = (e_k, k = 1, \dots, n)$ . We can use it to reconstruct the observed image  $x$  by minimizing  $\|x - F(x_0, e)\|$  over  $x_0$  and  $e$ . As a simple approximation, we can set  $e_k = 0$ , and write  $x_K = F(x_0)$ .

The advantage of short-run MCMC is that it can be conveniently scaled up to big datasets. After learning, it can be conveniently applied to testing data.

## 12.3 Equivalence between Discriminative and Descriptive Models

### 12.3.1 Discriminative Model

Let  $x$  be an input example, e.g., an image or a text, and let  $y$  be a label or annotation of  $x$ , e.g., the category that  $x$  belongs to in the case of classification. Let us focus on the classification problem, and suppose there are  $C$  categories. The commonly used soft-max classifier assumes that

$$p_\theta(y = c|x) = \frac{\exp(f_{c,\theta}(x))}{\sum_{c'=1}^C \exp(f_{c',\theta}(x))}, \quad (12.30)$$

where  $f_{c,\theta}$  is a deep network, and  $\theta$  denotes all the weight and bias parameters. For different  $c$ , the networks  $f_{c,\theta}$  may share a common body and only differ in head layer.

We can write the above model as

$$p_\theta(y = c|x) = \frac{1}{Z_\theta(x)} \exp(f_{c,\theta}(x)), \quad (12.31)$$

where

$$Z_\theta(x) = \sum_{c=1}^C \exp(f_{c,\theta}(x)). \quad (12.32)$$

The discriminative model  $p_\theta(y|x)$  can be learned by maximum likelihood. The log-likelihood is the average of

$$\log p_\theta(y|x) = f_{y,\theta}(x) - \log Z_\theta(x), \quad (12.33)$$

where the average is over the training set  $\{x, y\}$ . The gradient of  $\log p_\theta(y|x)$  with respect to  $\theta$  is

$$\nabla_\theta \log p_\theta(y|x) = \nabla_\theta f_{y,\theta}(x) - \mathcal{E}_{p_\theta(y|x)}[\nabla_\theta f_{y,\theta}(x)], \quad (12.34)$$

where

$$\nabla_\theta \log Z_\theta(x) = \mathcal{E}_{p_\theta(y|x)}[\nabla_\theta f_{y,\theta}(x)]. \quad (12.35)$$

Let  $p_{\text{data}}(x, y)$  be the data distribution of  $(x, y)$ . The MLE minimizes  $\text{KL}(p_{\text{data}}(y|x) \| p_\theta(y|x))$ , where for two conditional distributions  $p(y|x)$  and  $q(y|x)$ , their KL-divergence is defined as

$$\text{KL}(p(y|x) \| q(y|x)) = \mathcal{E}_{p(x,y)} \left[ \log \frac{p(y|x)}{q(y|x)} \right],$$

where the expectation is with respect to  $p(x, y) = p(x)p(y|x)$ , i.e., we also average over  $p(x)$  in addition to  $p(y|x)$ .

The above calculations are analogous to the calculations for the descriptive model. The difference is that for the discriminative model, the normalizing constant  $Z$  and the expectation are summations over  $y$ , where  $y$  belongs to a finite set of categories, whereas for the descriptive model, the normalizing constant  $Z$  and the expectation are integral over  $x$ , where  $x$  belongs to a high dimensional space. As a result, the expectation in descriptive model cannot be calculated in closed form, and has to be approximated by MCMC sampling such as Langevin dynamics.

A special case is binary classification, where  $y \in \{0, 1\}$ . It is usually assumed that

$$f_{0,\theta}(x) = 0, \quad f_{1,\theta}(x) = f_\theta(x), \quad (12.36)$$

so that

$$p_\theta(y = 1|x) = \frac{1}{1 + \exp(-f_\theta(x))} = \text{sigmoid}(f_\theta(x)), \quad (12.37)$$

and  $y$  follows a non-linear logistic regression on  $x$ .

### 12.3.2 Descriptive Model as Exponential Tilting of a Reference Distribution

A more general version of the descriptive model is of the following form of exponential tilting of a reference distribution [31, 231]

$$p_\theta(x) = \frac{1}{Z_\theta} \exp(f_\theta(x))q(x), \quad (12.38)$$

where  $q(x)$  is a given reference measure, such as uniform measure or Gaussian white noise distribution. The original form of the descriptive model corresponds to  $q(x)$  being a uniform measure. If  $q(x)$  is a Gaussian white noise distribution, then the energy function is  $-f_\theta(x) + \|x\|^2/2$ .

Although  $q(x)$  is usually taken to be a simple known distribution,  $q(x)$  can also be a model in its own right. We may call it a base model or a backbone model, and  $p_\theta(x)$  can be considered a correction of  $q(x)$ , where  $f_\theta(x)$  is the correction term. We may also call  $p_\theta(x)$  the energy-based correction of the base model  $q(x)$ .

### 12.3.3 Discriminative Model via Bayes Rule

The above exponential tilting leads to the following discriminative model. We can treat  $p_\theta$  as the positive distribution, and  $q(x)$  the negative distribution. Let  $y \in \{0, 1\}$ , and the prior probability  $p(y = 1) = \rho$ , so that  $p(y = 0) = 1 - \rho$ . Let  $p(x|y = 1) = p_\theta(x)$ ,  $p(x|y = 0) = q(x)$ . Then according to the Bayes rule [31, 111, 126, 133, 213, 231],

$$p(y = 1|x) = \frac{\exp(f_\theta(x) + b)}{1 + \exp(f_\theta(x) + b)}, \quad (12.39)$$

where  $b = \log(\rho/(1 - \rho)) - \log Z_\theta$ . This leads to a non-linear logistic regression. Sometimes, people call  $f_\theta(x) + b$  the logit or logit score, because

$$\log \frac{p(y = 1|x)}{p(y = 0|x)} = \text{logit}(p(y = 1|x)) = f_\theta(x) + b. \quad (12.40)$$

More generally, suppose we have  $C$  categories, and

$$p_{c,\theta}(x) = \frac{1}{Z_{c,\theta}} \exp(f_{c,\theta}(x))q(x), \quad c = 1, \dots, C, \quad (12.41)$$



where  $(f_{c,\theta}(x), c = 1, \dots, C)$  are  $C$  networks that may share the same body but with different heads. Suppose the prior probability for category  $c$  is  $\rho_c$ , then

$$p(y = c|x) = \frac{\exp(f_{c,\theta}(x) + b_c)}{\sum_{c=1}^C \exp(f_{c,\theta}(x) + b_c)}, \quad (12.42)$$

where  $b_c = \log \rho_c - \log Z_{c,\theta}$ . The above is a conventional soft-max classifier. Conversely, if  $p(y = c|x)$  is of the above form of soft-max classifier, then  $p_{c,\theta}(x)$  is of the form of exponential tilting based on the logit score  $f_{c,\theta}(x) + b_c$ . Thus the discriminative model and the descriptive model are equivalent to each other.

### 12.3.4 Noise Contrastive Estimation and Introspective Learning

The above equivalence suggests that we can learn descriptive model by fitting a logistic regression. Specifically, suppose we want to learn a descriptive model  $p_\theta(x) = \frac{1}{Z_\theta} \exp(f_\theta(x))q(x)$ , where  $q(x)$  is a noise distribution, such as Gaussian white noise distribution. We can treat the observed examples as the positive examples, so that for each positive  $x$ ,  $y = 1$ , and we generate negative examples from the noise distribution  $q(x)$ , so that for each negative example  $x \sim q(x)$ ,  $y = 0$ . Then we learn a discriminator in the form of logistic regression to distinguish between the positive and negative examples, then  $\text{logit}(p(y = 1|x)) = f_\theta(x) + b$ , where  $b = \log(\rho/(1 - \rho)) - \log Z_\theta$ , where  $\rho$  is the proportion of the positive examples. We can learn both  $\theta$  and  $b$  by fitting a logistic regression, where  $b$  is treated as an independent bias or intercept term, even though  $\log Z_\theta$  depends on  $\theta$ . This enables us to learn  $f_\theta$  and estimate  $\log Z_\theta$ . This is called noise contrastive estimation.

The problem with the above scheme is that the noise distribution and the data distribution usually do not have much overlap, especially if  $x$  is of high dimensionality. As a result,  $f_\theta(x)$  cannot be well learned.

The introspective learning method tries to remedy the above problem with sampling. After learning  $f_\theta(x)$  by noise contrastive estimation, we want to inspect whether  $f_\theta(x)$  is well learned. We then treat the current  $p_\theta(x)$  as our new  $q(x)$ , and we draw negative samples from it. If it is well learned, then the negative samples will be close to the positive examples. To check that, we fit a logistic regression again on the positive examples and negative examples from the new  $q(x)$ . Then we learn a new  $\Delta f_\theta(x)$  by the new logistic regression. This  $\Delta f_\theta(x)$  can then be added to the previous learned  $f_\theta(x)$  to obtain the new  $f_\theta(x)$ . We can keep repeating this process until  $\Delta f_\theta(x)$  is small. This scheme is called introspective learning. It is related to GAN, except that  $f_\theta(x)$  plays the roles of both discriminator and generator.

## 12.4 Generative Model

### 12.4.1 Model and Origin

Both discriminative model and descriptive model are based on a bottom-up network  $f_\theta(x)$ . The generative model is based on top-down network with latent variables. The prototype of such a model is factor analysis. Let  $x$  be the observed example, which is a  $D$ -dimensional vector. We assume that  $x$  can be explained by a  $d$ -dimensional latent vector, each element of which is called a factor. Given  $z$ ,  $x$  is generated by  $x = Wz + \epsilon$ , where  $W$  is a  $D \times d$  matrix, sometimes called loading matrix. It is usually assumed that  $z \sim N(0, I_d)$ , where  $I_d$  is  $d$ -dimensional identity matrix,  $\epsilon \sim N(0, \sigma^2 I_D)$ , and  $\epsilon$  is independent of  $z$ . The factor analysis model originated from psychometrics, where  $x$  consists of a pupil's scores on a number of subjects, and  $z = (z_1, z_2)$ , where  $z_1$  is verbal intelligence, and  $z_2$  is analytical intelligence.

By generalizing the prior assumption  $z \sim N(0, I_d)$ , the factor analysis model has been generalized to independent component analysis, sparse coding, non-negative matrix factorization, as well as matrix factorization for recommender system.



A recent generalization [85] is to keep the prior assumption about  $z$ , but replace the linear model  $x = Wz + \epsilon$  by a non-linear model  $x = g_\theta(z) + \epsilon$ , where  $g_\theta(z)$  is parametrized by a top-down neural network where  $\theta$  collects all the weight and bias parameters. In the case of image modeling,  $g_\theta(z)$  is usually a convolutional neural network, which is sometimes called deconvolutional network, due to its top-down nature. The above model leads to a conditional or generation model  $p_\theta(x|z)$ , such that

$$\log p_\theta(x, z) = \log[p(z)p_\theta(x|z)] \quad (12.43)$$

$$= -\frac{1}{2} [\|z\|^2 + \|x - g_\theta(z)\|^2/\sigma^2] + c, \quad (12.44)$$

where  $c$  is a constant independent of  $\theta$ .  $\sigma^2$  is usually treated as a tuning parameter.

The joint distribution of  $(x, z)$  is  $p_\theta(x, z) = p(z)p_\theta(x|z)$ . The marginal distribution of  $x$  is  $p_\theta(x) = \int p_\theta(x, z)dz$ . Given  $x$ , the inference of  $z$  can be based on the posterior distribution  $p_\theta(z|x) = p_\theta(x, z)/p_\theta(x)$ .

The above model is often referred to as the generator network in the literature.

### 12.4.2 Generative Model with Multi-layer Latent Variables

While it is computationally convenient to have a single latent noise vector at the top layer, it does not account for the fact that patterns can appear at multiple layers of compositions or abstractions (e.g., face  $\rightarrow$  (eyes, nose, mouth)  $\rightarrow$  (edges, corners)  $\rightarrow$  pixels), where variations and randomness occur at multiple layers. To capture such a hierarchical structure, it is desirable to introduce multiple layers of latent variables organized in a top-down architecture [167]. Specifically, we have  $z = (z_l, l = 1, \dots, L)$ , where layer  $L$  is the top layer, and layer 1 is the bottom layer above  $x$ . For notational simplicity, we let  $x = z_0$ . We can then specify  $p_\theta(z)$  as

$$p_\theta(z) = p_\theta(z_L) \prod_{l=0}^{L-1} p_\theta(z_l|z_{l+1}). \quad (12.45)$$

One concrete example is  $z_L \sim N(0, I)$ ,  $[z_l|z_{l+1}] \sim N(\mu_l(z_{l+1}), \sigma_l^2(z_{l+1}))$ ,  $l = 0, \dots, L-1$ . where  $\mu_l()$  and  $\sigma_l^2()$  are the mean vector and the diagonal variance-covariance matrix of  $z_l$  respectively, and they are functions of  $z_{l+1}$ .  $\theta$  collects all the parameters in these functions.  $p_\theta(x, z)$  can be obtained similarly as in Equation 12.44.

### 12.4.3 MLE Learning and Posterior Inference

Let  $p_{\text{data}}(x)$  be the data distribution that generates the example  $x$ . The learning of parameters  $\theta$  of  $p_\theta(x)$  can be based on  $\min_\theta \text{KL}(p_{\text{data}}(x)||p_\theta(x))$ . If we observe training examples  $\{x_i, i = 1, \dots, n\} \sim p_{\text{data}}(x)$ , the above minimization can be approximated by maximizing the log-likelihood

$$L(\theta) = \frac{1}{n} \sum_{i=1}^n \log p_\theta(x_i) \doteq \mathcal{E}_{p_{\text{data}}}[\log p_\theta(x)], \quad (12.46)$$

which leads to MLE.

The gradient of the log-likelihood,  $L'(\theta)$ , can be computed according to the following identity:

$$\delta_\theta(x) = \nabla_\theta \log p_\theta(x) = \frac{1}{p_\theta(x)} \nabla_\theta p_\theta(x) = \frac{1}{p_\theta(x)} \int \nabla_\theta p_\theta(x, z) dz = \mathcal{E}_{p_\theta(z|x)} [\nabla_\theta \log p_\theta(x, z)]. \quad (12.47)$$

Thus

$$L'(\theta) = \mathcal{E}_{p_{\text{data}}}[\delta_{\theta}(x)] = \mathcal{E}_{p_{\text{data}}(x)}\mathcal{E}_{p_{\theta}(z|x)}[\nabla_{\theta} \log p_{\theta}(x, z)] \quad (12.48)$$

$$\doteq \frac{1}{n} \sum_{i=1}^n \mathcal{E}_{p_{\theta}(z_i|x_i)}[\nabla_{\theta} \log p_{\theta}(x_i, z_i)]. \quad (12.49)$$

The expectation with respect to  $p_{\theta}(z|x)$  can be approximated by Monte Carlo samples. Each learning iteration updates  $\theta$  by  $\theta_{t+1} = \theta_t + \eta_t L'(\theta_t)$ .

#### 12.4.4 Posterior Sampling

Sampling from  $p_{\theta}(z|x)$  usually requires MCMC. One convenient MCMC is Langevin dynamics, which iterates

$$z_{t+1} = z_t + s \nabla_z \log p_{\theta}(z_t|x) + \sqrt{2s} e_t, \quad (12.50)$$

where  $e_t \sim \mathcal{N}(0, I)$ ,  $t$  indexes the time step of the Langevin dynamics, and  $s$  is the step size. The Langevin dynamics consists of a gradient descent term on  $-\log p(z|x)$ . In the case of generator network, it amounts to gradient descent on  $\|z\|^2/2 + \|x - g_{\theta}(z)\|^2/2\sigma^2$ , which is penalized reconstruction error. The Langevin dynamics also consists of a white noise diffusion term  $\sqrt{2s}e_t$  to create randomness for sampling from  $p_{\theta}(z|x)$ .

For small step size  $s$ , the marginal distribution of  $z_t$  will converge to  $p_{\theta}(z|x)$  as  $t \rightarrow \infty$  regardless of the initial distribution of  $z_0$ . More specifically, let  $p_t(z)$  be the marginal distribution of  $z_t$  of the Langevin dynamics, then  $\text{KL}(p_t(z)||p_{\theta}(z|x))$  decreases monotonically to 0, that is, by increasing  $t$ , we reduce  $\text{KL}(p_t(z)||p_{\theta}(z|x))$  monotonically.

#### 12.4.5 Perturbation of KL-divergence

Again we understand the MLE learning algorithm by perturbing the KL-divergence for MLE. Define  $D(\theta) = \text{KL}(p_{\text{data}}||p_{\theta})$ . It is the objective function of MLE. Let  $\theta_t$  be the estimate at iteration  $t$ . Let us consider the following perturbation of  $D(\theta)$ ,

$$S(\theta) = D(\theta) + \text{KL}(p_{\theta_t}(z|x)||p_{\theta}(z|x)) \quad (12.51)$$

$$= \text{KL}(p_{\text{data}}(x)||p_{\theta}(x)) + \text{KL}(p_{\theta_t}(z|x)||p_{\theta}(z|x)) \quad (12.52)$$

$$= \text{KL}(p_{\text{data}, \theta_t}(x, z)||p_{\theta}(x, z)), \quad (12.53)$$

where we define  $p_{\text{data}, \theta_t}(x, z) = p_{\text{data}}(x)p_{\theta_t}(z|x)$ . Again  $S(\theta)$  is a surrogate for  $D(\theta)$  at  $\theta_t$ , and  $S(\theta)$  is simpler than  $D(\theta)$  because  $S(\theta)$  is based on the joint distributions instead of the marginal distributions as in  $D(\theta)$ . Unlike the joint distribution  $p_{\theta}(x, z) = p(z)p_{\theta}(x|z)$ , the marginal distribution  $p_{\theta}(x) = \int p_{\theta}(x, z)dz$  is implicit as it is an intractable integral.

The perturbation term  $\text{KL}(p_{\theta_t}(z|x)||p_{\theta}(z|x))$ , as a function of  $\theta$ , achieves its minimum 0 at  $\theta = \theta_t$ , and its derivative at  $\theta = \theta_t$  is zero. Thus  $S(\theta)$  and  $D(\theta)$  touches each other at  $\theta_t$ , and they share the same gradient at  $\theta_t$ .

$$-S(\theta) = \mathcal{E}_{p_{\text{data}}(x)}\mathcal{E}_{p_{\theta_t}(z|x)}[\log p_{\theta}(x, z)] - \text{entropy}(p_{\text{data}, \theta_t}(x, z)). \quad (12.54)$$

$$-S'(\theta) = \mathcal{E}_{p_{\text{data}}(x)}\mathcal{E}_{p_{\theta_t}(z|x)}[\nabla_{\theta} \log p_{\theta}(x, z)]. \quad (12.55)$$

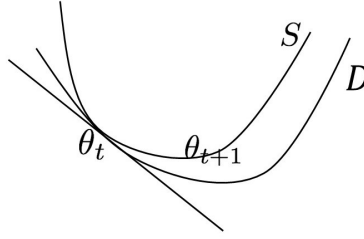


Figure 12.8: Reprinted with permission from [86]. The surrogate  $S$  majorizes (upper bounds)  $D$ , and they touch each other at  $\theta_t$  with the same tangent.

Thus, the learning gradient at  $\theta_t$  is

$$-D'(\theta_t) = -S'(\theta_t) = \mathcal{E}_{p_{\text{data}}}[ \delta_{\theta_t}(x) ] = \mathcal{E}_{p_{\text{data}}(x)} \mathcal{E}_{p_{\theta_t}(z|x)} [ \nabla_{\theta} \log p_{\theta_t}(x, z) ]. \quad (12.56)$$

This provides another justification of the learning algorithm.

The above perturbation of KL-divergence can be compared to that in the descriptive model, where the sign in front of the second KL-divergence is negative, in order to cancel the intractable  $\log Z_{\theta}$  term. For generative model, the sign in front of the second KL-divergence is positive, in order to change the marginal distributions in the first KL-divergence, i.e.,  $D(\theta)$ , into the joint distributions, so that  $p_{\theta}(z, x) = p(z)p_{\theta}(x|z)$  is obtained in closed form.

#### 12.4.6 EM Algorithm

The above perturbation underlies the EM algorithm, where, instead of gradient descent, we let  $\theta_{t+1} = \arg \min_{\theta} S(\theta)$ . EM stands for expectation-maximization. The expectation is about computing  $S(\theta)$ , and the maximization is about maximizing  $-S(\theta)$ .

Recall that  $S(\theta)$  and  $D(\theta)$  touch each other at  $\theta_t$ , and  $S(\theta)$  is above  $D(\theta)$ , i.e.,  $S(\theta) \geq D(\theta)$ . Sometimes it is said  $S(\theta)$  majorizes  $D(\theta)$ . Thus if we reduce  $S(\theta)$  by the above minimization, so that  $S(\theta_{t+1}) \leq S(\theta_t)$ , we will reduce  $D(\theta)$  even more so that we also have  $D(\theta_{t+1}) \leq D(\theta_t)$ . This underlies the monotonicity of the EM algorithm.

#### 12.4.7 Noise-initialized Short-run MCMC as an Approximate Inference Engine

It is impractical to run long chains to sample from  $p_{\theta}(z|x)$ . We thus propose the following short run MCMC as inference dynamics [167], with a fixed small  $K$  (e.g.,  $K = 25$ ),

$$z_0 \sim p(z), z_{k+1} = z_k + s \nabla_z \log p_{\theta}(z_k|x) + \sqrt{2s} e_k, \quad k = 1, \dots, K, \quad (12.57)$$

where  $p(z)$  is the prior noise distribution of  $z$ .

We can write the above short run MCMC as

$$z_0 \sim p(z), \quad z_{k+1} = z_k + s R(z_k) + \sqrt{2s} e_k, \quad k = 1, \dots, K, \quad (12.58)$$

$R(z) = \nabla_z \log p_{\theta}(z|x)$ , where we omit  $x$  and  $\theta$  in  $R(z)$  for simplicity of notation.

To further simplify the notation, we may write the short run MCMC as

$$z_0 \sim p(z), \quad z_K = F(z_0, e), \quad (12.59)$$

where  $e = (e_k, k = 1, \dots, K)$ , and  $F$  composes the  $K$  steps of Langevin updates.

Let the distribution of  $z_K$  be  $\tilde{p}(z)$ . Recall that the distribution of  $z_K$  also depends on  $x$  and  $\theta$ , and step size  $s$ , so that in full notation, we may write  $\tilde{p}(z)$  as  $\tilde{p}_{s,\theta}(z|x)$ .

For each  $x$ , we define

$$\tilde{\delta}_\theta(x) = \mathcal{E}_{\tilde{p}_{s,\theta}(z|x)} [\nabla_\theta \log p_\theta(x, z)], \quad (12.60)$$

and modify the learning algorithm to

$$\theta_{t+1} = \theta_t + \eta_t \mathcal{E}_{p_{\text{data}}} [\tilde{\delta}_{\theta_t}(x)] = \theta_t + \eta_t E_{p_{\text{data}}} \mathcal{E}_{\tilde{p}_{s,\theta_t}(z|x)} [\nabla_\theta \log p_{\theta_t}(x, z)], \quad (12.61)$$

where  $\eta_t$  is the learning rate,  $\mathcal{E}_{\tilde{p}_{s,\theta_t}(z|x)}$  (here we use the full notation  $\tilde{p}_{s,\theta}(z|x)$  instead of the abbreviated notation  $q(z)$ ) can be approximated by sampling from  $\tilde{p}_{s,\theta_t}(z|x_i)$  using the noise initialized  $K$ -step Langevin dynamics.

Compared to MLE learning algorithm, we replace  $p_\theta(z|x)$  by  $\tilde{p}_{s,\theta}(z|x)$ , and fair Monte Carlo samples from  $\tilde{p}_{s,\theta}(z|x)$  can be obtained by short run MCMC.

One major advantage of the proposed method is that it is simple and automatic. For models with multiple layers of latent variables that may be organized in complex top-down architectures, the gradient computation in Langevin dynamics is automatic on modern deep learning platforms. Such dynamics naturally integrates explaining-away competitions and bottom-up and top-down interactions between multiple layers of latent variables. It thus enables researchers to explore flexible generative models without dealing with the challenging task of designing and learning the inference models.

One class of generative models that are of particular interest are biologically plausible models, such as Boltzmann machine and the generation model of the Helmholtz machine, where each node is a latent variable. With such a large number of latent variables, designing an inference network to regulate the bottom-up and top-down flows of information as well as lateral inhibitions becomes a daunting task. However, short run MCMC is automatic, natural, and biologically plausible as it may be related to attractor dynamics [6, 97, 181].

In computational neuroscience, the dynamics of the neuron activities is often modeled by attractor dynamics [6, 97, 181]. However, the objective function of the attractor dynamics is often implicit, thus it is unclear what the computational problem is that the attractor dynamics is solving. For the attractor dynamics to be implemented in real time, the dynamics is necessarily a short run dynamics. Our short run MCMC is guided by a top-down model with a well-defined posterior distribution of the latent variables. It may be connected to the attractor dynamics and help us understand the latter.

#### 12.4.8 Objective function and estimating equation

The following are justifications of learning with short-run MCMC.

(1) **Objective function** Again we use perturbation of KL-divergence. At iteration  $t$ , with  $\theta_t$  fixed, the learning algorithm follows the gradient of the following perturbation of  $D(\theta)$  at  $\theta = \theta_t$ :

$$S(\theta) = D(\theta) + \text{KL}(\tilde{p}_{s,\theta_t}(z|x) \| p_\theta(z|x)) = \text{KL}(p_{\text{data}}(x) \| p_\theta(x)) + \text{KL}(\tilde{p}_{s,\theta_t}(z|x) \| p_\theta(z|x)), \quad (12.62)$$

so that  $\theta_{t+1} = \theta_t - \eta_t S'(\theta_t)$ , where  $\eta_t$  is the step size, and

$$-S'(\theta) = \mathcal{E}_{p_{\text{data}}(x)} \mathcal{E}_{\tilde{p}_{s,\theta_t}(z|x)} [\nabla_\theta \log p_\theta(x, z)]. \quad (12.63)$$

$$-S'(\theta_t) = \mathcal{E}_{p_{\text{data}}} [\tilde{\delta}_{\theta_t}(x)] = \mathcal{E}_{p_{\text{data}}(x)} \mathcal{E}_{\tilde{p}_{s,\theta_t}(z|x)} [\nabla_\theta \log p_{\theta_t}(x, z)]. \quad (12.64)$$

Compared to the perturbation of KL-divergence in MLE learning, we use  $\tilde{p}_{s,\theta_t}(z|x)$  instead of  $p_{\theta_t}(z|x)$ . While sampling  $p_{\theta_t}(z|x)$  can be impractical if it is multi-modal, sampling  $\tilde{p}_{s,\theta_t}(z|x)$  is practical because it is a short-run MCMC.

(2) **Estimating equation.** The fixed point of the learning algorithm (12.61) solves the following estimating equation:

$$\frac{1}{n} \sum_{i=1}^n \mathcal{E}_{\tilde{p}_{s,\theta}(z_i|x_i)} [\nabla_{\theta} \log p_{\theta}(x_i, z_i)] \doteq \mathcal{E}_{p_{\text{data}}(x)} \mathcal{E}_{\tilde{p}_{s,\theta}(z|x)} [\nabla_{\theta} \log p_{\theta}(x, z)] = 0. \quad (12.65)$$

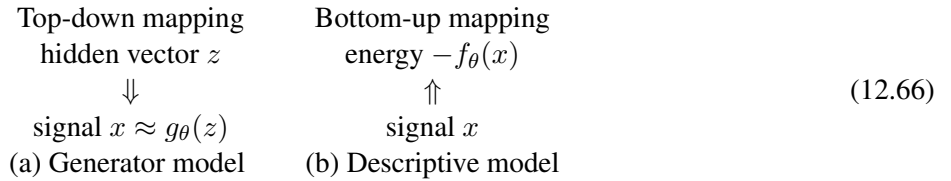
If we approximate  $\mathcal{E}_{\tilde{p}_{s,\theta}(z_i|x_i)}$  by Monte Carlo samples from  $\tilde{p}_{s,\theta_t}(z_i|x_i)$ , then the learning algorithm becomes Robbins-Monro algorithm for stochastic approximation.

The bias of the learned  $\theta$  based on short run MCMC relative to the MLE depends on the gap between  $\tilde{p}_{s,\theta}(z|x)$  and  $p_{\theta}(z|x)$ . We suspect that this bias may actually be beneficial in the following sense. The learning gradient seeks to decrease  $D(\theta)$  while decreasing  $\text{KL}(\tilde{p}_{s,\theta_t}(z_i|x_i) \| p_{\theta}(z_i|x_i))$ . The latter tends to bias the learned model so that its posterior distribution  $p_{\theta}(z_i|x_i)$  is close to the short run MCMC  $\tilde{p}_{s,\theta_t}(z_i|x_i)$ , i.e., our learning method may bias the model to make inference by short run MCMC accurate.

We can optimize the step size  $s$  of the short-run Langevin dynamics by minimizing  $\text{KL}(\tilde{p}_{s,\theta_t}(z_i|x_i) \| p_{\theta}(z_i|x_i))$  over  $s$ . This is a variational optimization.

## 12.5 Descriptive Model in Latent Space of Generative Model

### 12.5.1 Top-down and Bottom-up



The above diagram compares the generative model and the descriptive model. The former is based on top-down generation, whereas the latter is based on bottom-up description.

The top-down model is a very natural representation of knowledge, with its multiple layers of latent variables representing concepts at multiple levels of abstractions. The top-down model is also called directed acyclic graphical model. It is characterized by independence or conditional independence assumptions of the latent variables. Such assumptions limit the expressive power of the top-down model.

For the special case of generator network, there is a latent vector  $z$  at the top layer, which generates the example  $x$  via the top-down generation mapping  $g_{\theta}(z)$ . The prior distribution of  $z$  is usually assumed to be a simple noise distribution, e.g., the Gaussian white noise distribution  $z \sim \mathcal{N}(0, I)$ . The top-down  $g_{\theta}(z)$  maps this simple isotropic unimodal prior distribution to the multi-modal data distribution  $p_{\text{data}}$ . However, the expressive power may be limited by the simple prior distribution  $p(z)$  (as well as the simple Gaussian white noise distribution of  $\epsilon$  in  $x = g_{\theta}(z) + \epsilon$ ). The marginal distribution of  $p_{\theta}(x) = \int p(z)p_{\theta}(x|z)dz$  is implicit because of the intractable integral over the latent  $z$ .

The bottom-up model only needs to specify a scalar-valued energy function  $-f_{\theta}(x)$ , instead of a vector-valued  $g_{\theta}(z)$ , while leaving the generative task to MCMC. It specifies the distribution  $p_{\theta}(x) = \frac{1}{Z_{\theta}} \exp(f_{\theta}(x))$  explicitly even though the normalizing constant  $Z_{\theta}$  is intractable. Compared to the generator model, the descriptive model tends to have stronger expressive power in terms of synthesis ability.

However, because  $p_{\text{data}}$  tends to be highly multi-modal, the learned  $p_{\theta}$  can also be highly multi-modal. As a result, MCMC sampling cannot mix. Even though we can use short-run MCMC to learn the model and synthesize images, the model is admittedly biased.

### 12.5.2 Descriptive Model in Latent Space

We follow the philosophy of empirical Bayes, that is, instead of assuming a given prior distribution for the latent vector, as in the generator network, we learn a prior model from empirical observations.

Specifically, we assume the latent vector follows a descriptive model or more specifically, an energy-based correction of the isotropic Gaussian white noise prior distribution. We call this model the latent space descriptive model. Such a model adds more expressive power to the generator model. In the latent space, the descriptive model is close to unimodal as it is a correction of the unimodal Gaussian distribution, MCMC sampling is expected to mix well.

The MLE learning of the generative model with a latent space descriptive model involves MCMC sampling of latent vector from both the prior and posterior distributions. Parameters of the prior model can then be updated based on the statistical difference between samples from the two distributions. Parameters of the top-down network can be updated based on the samples from the posterior distribution as well as the observed data.

### 12.5.3 Model

Let  $x \in \mathbb{R}^D$  be an observed example such as an image or a piece of text, and let  $z \in \mathbb{R}^d$  be the latent variables, where  $D \gg d$ . The joint distribution of  $(x, z)$  is

$$p_{\theta}(x, z) = p_{\alpha}(z)p_{\beta}(x|z), \quad (12.67)$$

where  $p_{\alpha}(z)$  is the prior model with parameters  $\alpha$ ,  $p_{\beta}(x|z)$  is the top-down generation model with parameters  $\beta$ , and  $\theta = (\alpha, \beta)$ .

The prior model  $p_{\alpha}(z)$  is formulated as a descriptive model or an energy-based model,

$$p_{\alpha}(z) = \frac{1}{Z(\alpha)} \exp(f_{\alpha}(z))p_0(z). \quad (12.68)$$

where  $p_0(z)$  is a reference distribution, assumed to be isotropic Gaussian white noise distribution.  $f_{\alpha}(z)$  is the negative energy and is parameterized by a small multi-layer perceptron with parameters  $\alpha$ .  $Z(\alpha) = \int \exp(f_{\alpha}(z))p_0(z)dz = \mathcal{E}_{p_0}[\exp(f_{\alpha}(z))]$  is the normalizing constant or partition function.

The prior model (12.68) can be interpreted as an energy-based correction or exponential tilting of the original prior distribution  $p_0$ , which is the prior distribution in the generator model.

The generation model is the same as the top-down network in the generator model. For image modeling,

$$x = g_{\beta}(z) + \epsilon, \quad (12.69)$$

where  $\epsilon \sim N(0, \sigma^2 I_D)$ , so that  $p_{\beta}(x|z) \sim N(g_{\beta}(z), \sigma^2 I_D)$ . Usually  $\sigma^2$  takes an assumed value. For text modeling, let  $x = (x^{(t)}, t = 1, \dots, T)$  where each  $x^{(t)}$  is a token. A commonly used model is to define  $p_{\beta}(x|z)$  as a conditional autoregressive model,

$$p_{\beta}(x|z) = \prod_{t=1}^T p_{\beta}(x^{(t)}|x^{(1)}, \dots, x^{(t-1)}, z) \quad (12.70)$$

which is often parameterized by a recurrent network with parameters  $\beta$ .

In the original generator model, the top-down network  $g_\beta$  maps the unimodal prior distribution  $p_0$  to be close to the usually highly multi-modal data distribution. The prior model in (12.68) refines  $p_0$  so that  $g_\beta$  maps the prior model  $p_\alpha$  to be closer to the data distribution. The prior model  $p_\alpha$  does not need to be highly multi-modal because of the expressiveness of  $g_\beta$ .

The marginal distribution is  $p_\theta(x) = \int p_\theta(x, z) dz = \int p_\alpha(z) p_\beta(x|z) dz$ . The posterior distribution is  $p_\theta(z|x) = p_\theta(x, z)/p_\theta(x) = p_\alpha(z) p_\beta(x|z)/p_\theta(x)$ .

In the above model, we exponentially tilt  $p_0(z)$ . We can also exponentially tilt  $p_0(x, z) = p_0(z) p_\beta(x|z)$  to  $p_\theta(x, z) = \frac{1}{Z(\theta)} \exp(f_\alpha(x, z)) p_0(x, z)$ . Equivalently, we may also exponentially tilt  $p_0(z, \epsilon) = p_0(z) p(\epsilon)$ , as the mapping from  $(z, \epsilon)$  to  $(z, x)$  is a change of variable. This leads to a descriptive model in both the latent space and data space, which makes learning and sampling more complex. Therefore, we choose to only tilt  $p_0(z)$  and leave  $p_\beta(x|z)$  as a directed top-down generation model.

#### 12.5.4 Maximum Likelihood

Suppose we observe training examples  $(x_i, i = 1, \dots, n)$ . The log-likelihood function is

$$L(\theta) = \frac{1}{n} \sum_{i=1}^n \log p_\theta(x_i) \doteq \mathcal{E}_{p_{\text{data}}} [\log p_\theta(x)]. \quad (12.71)$$

The learning gradient can be calculated according to

$$\nabla_\theta \log p_\theta(x) = \mathcal{E}_{p_\theta(z|x)} [\nabla_\theta \log p_\theta(x, z)] = \mathcal{E}_{p_\theta(z|x)} [\nabla_\theta (\log p_\alpha(z) + \log p_\beta(x|z))]. \quad (12.72)$$

For the prior model,  $\nabla_\alpha \log p_\alpha(z) = \nabla_\alpha f_\alpha(z) - \mathcal{E}_{p_\alpha(z)} [\nabla_\alpha f_\alpha(z)]$ . Thus the learning gradient for an example  $x$  is

$$\delta_\alpha(x) = \nabla_\alpha \log p_\theta(x) = \mathcal{E}_{p_\theta(z|x)} [\nabla_\alpha f_\alpha(z)] - \mathcal{E}_{p_\alpha(z)} [\nabla_\alpha f_\alpha(z)]. \quad (12.73)$$

The above equation has an empirical Bayes nature.  $p_\theta(z|x)$  is based on the empirical observation  $x$ , while  $p_\alpha$  is the prior model.  $\alpha$  is updated based on the difference between  $z$  inferred from empirical observation  $x$ , and  $z$  sampled from the current prior.

For the generation model,

$$\delta_\beta(x) = \nabla_\beta \log p_\theta(x) = \mathcal{E}_{p_\theta(z|x)} [\nabla_\beta \log p_\beta(x|z)], \quad (12.74)$$

where  $\log p_\beta(x|z) = -\|x - g_\beta(z)\|^2/(2\sigma^2) + \text{constant}$  or  $\sum_{t=1}^T \log p_\beta(x^{(t)}|x^{(1)}, \dots, x^{(t-1)}, z)$  for image and text modeling respectively, which is about the reconstruction error.

Writing  $\delta_\theta(x) = (\delta_\alpha(x), \delta_\beta(x))$ , we have  $L'(\theta) = \mathcal{E}_{p_{\text{data}}} [\delta_\theta(x)]$ , and the learning algorithm is  $\theta_{t+1} = \theta_t + \eta_t \mathcal{E}_{p_{\text{data}}} [\delta_\theta(x)]$ .

Expectations in (12.73) and (12.74) require MCMC sampling of the prior model  $p_\alpha(z)$  and the posterior distribution  $p_\theta(z|x)$ . We can use Langevin dynamics. For a target distribution  $\pi(z)$ , the dynamics iterates

$$z_{k+1} = z_k + s \nabla_z \log \pi(z_k) + \sqrt{2s} e_k, \quad (12.75)$$

where  $k$  indexes the time step of the Langevin dynamics,  $s$  is a small step size, and  $e_k \sim N(0, I_d)$  is the Gaussian white noise.  $\pi(z)$  can be either  $p_\alpha(z)$  or  $p_\theta(z|x)$ . In either case,  $\nabla_z \log \pi(z)$  can be efficiently computed by back-propagation.

### 12.5.5 Short-run MCMC for Synthesis and Inference

Convergence of Langevin dynamics to the target distribution requires infinite steps with infinitesimal step size, which is impractical. We thus propose to use short-run MCMC for approximate sampling. This is in agreement with the philosophy of variational inference, which accepts the intractability of the target distribution and seeks to approximate it by a simpler distribution. The difference is that we adopt short-run Langevin dynamics instead of learning a separate network for approximation.

The short-run Langevin dynamics is always initialized from the fixed initial distribution  $p_0$ , and only runs a fixed number of  $K$  steps, e.g.,  $K = 20$ ,

$$z_0 \sim p_0(z), \quad z_{k+1} = z_k + s \nabla_z \log \pi(z_k) + \sqrt{2s} \epsilon_k, \quad k = 1, \dots, K. \quad (12.76)$$

Denote the distribution of  $z_K$  to be  $\tilde{\pi}(z)$ . Because of fixed  $p_0(z)$  and fixed  $K$  and  $s$ , the distribution  $\tilde{\pi}$  is well defined. In this section, we put  $\sim$  sign on top of the symbols to denote distributions or quantities produced by short-run MCMC, and for simplicity, we omit the dependence on  $K$  and  $s$  in notation. The Kullback-Leibler divergence  $\text{KL}(\tilde{\pi}||\pi)$  decreases to zero monotonically as  $K \rightarrow \infty$ .

Specifically, denote the distribution of  $z_K$  to be  $\tilde{p}_\alpha(z)$  if the target  $\pi(z) = p_\alpha(z)$ , and denote the distribution of  $z_K$  to be  $\tilde{p}_\theta(z|x)$  if  $\pi(z) = p_\theta(z|x)$ . We can then replace  $p_\alpha(z)$  by  $\tilde{p}_\alpha(z)$  and replace  $p_\theta(z|x)$  by  $\tilde{p}_\theta(z|x)$  in equations (12.73) and (12.74), so that the learning gradients in equations (12.73) and (12.74) are modified to

$$\tilde{\delta}_\alpha(x) = \mathcal{E}_{\tilde{p}_\theta(z|x)}[\nabla_\alpha f_\alpha(z)] - \mathcal{E}_{\tilde{p}_\alpha(z)}[\nabla_\alpha f_\alpha(z)], \quad (12.77)$$

$$\tilde{\delta}_\beta(x) = \mathcal{E}_{\tilde{p}_\theta(z|x)}[\nabla_\beta \log p_\beta(x|z)]. \quad (12.78)$$

We then update  $\alpha$  and  $\beta$  based on (12.77) and (12.78), where the expectations can be approximated by Monte Carlo samples. Specifically, writing  $\tilde{\delta}_\theta(x) = (\tilde{\delta}_\alpha(x), \tilde{\delta}_\beta(x))$ , the learning algorithm is  $\theta_{t+1} = \theta_t + \eta_t \mathcal{E}_{p_{\text{data}}}[\tilde{\delta}_\theta(x)]$ .

The short-run MCMC sampling is always initialized from the same initial distribution  $p_0(z)$ , and always runs a fixed number of  $K$  steps. This is the case for both training and testing stages, which share the same short-run MCMC sampling.

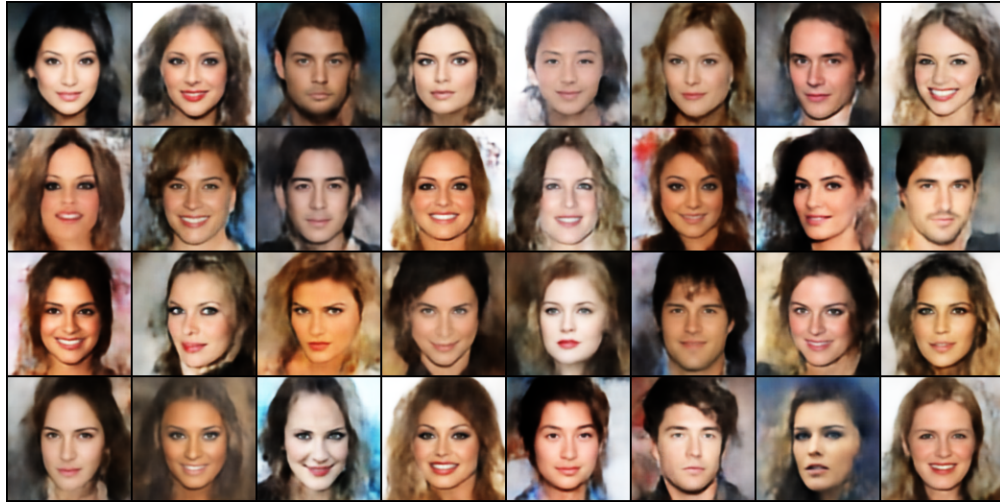


Figure 12.9: Reprinted with permission from [175]. Generated images for CelebA ( $128 \times 128 \times 3$ ).



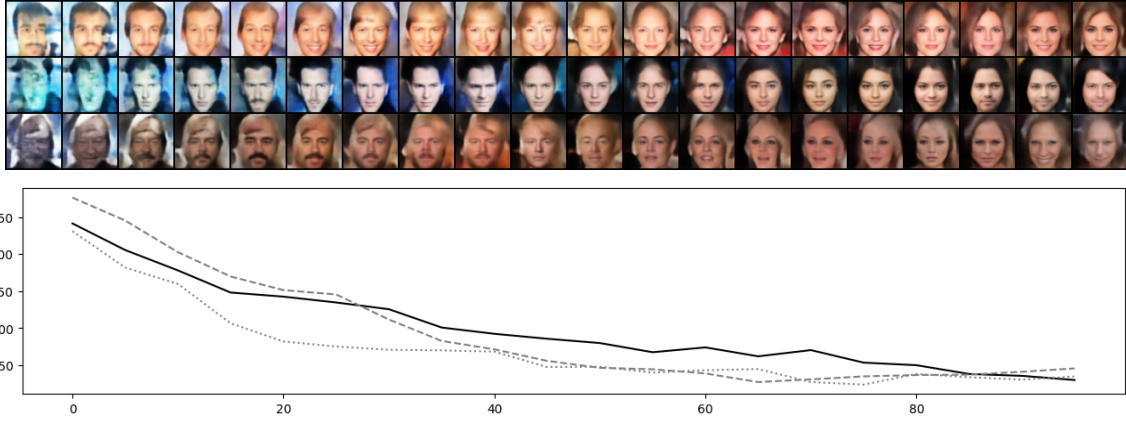


Figure 12.10: Reprinted with permission from [175]. Transition of Markov chains initialized from  $p_0(z)$  towards  $\tilde{p}_\alpha(z)$  for 100 Langevin dynamics steps. *Top*: Trajectory in the CelebA data-space. *Bottom*: Energy profile over time.

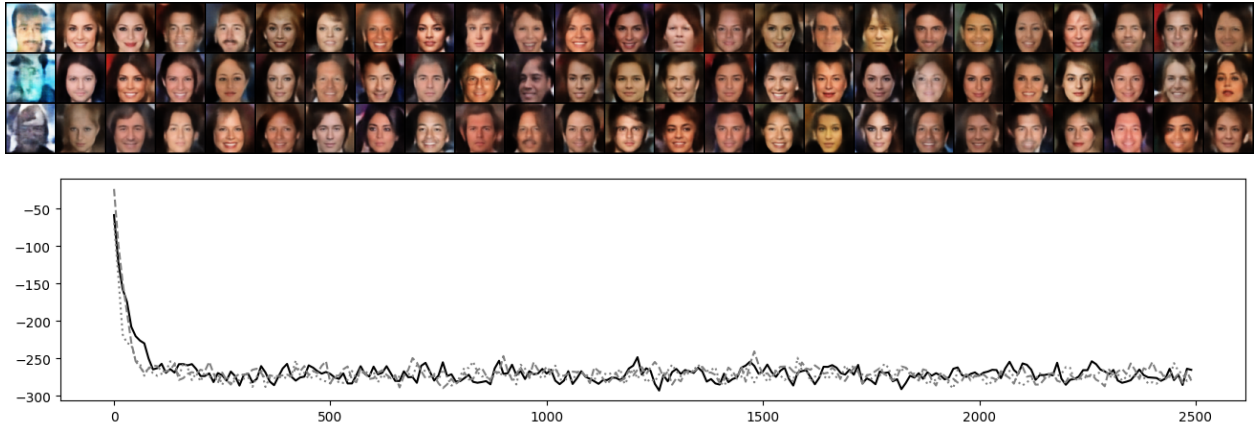


Figure 12.11: Reprinted with permission from [175]. Transition of Markov chains initialized from  $p_0(z)$  towards  $\tilde{p}_\alpha(z)$  for 2, 500 Langevin dynamics steps. *Top*: Trajectory in the CelebA data-space for every 100 steps. *Bottom*: Energy profile over time.

### 12.5.6 Divergence Perturbation

The learning algorithm based on short-run MCMC sampling is a modification or perturbation of maximum likelihood learning, where we replace  $p_\alpha(z)$  and  $p_\theta(z|x)$  by  $\tilde{p}_\alpha(z)$  and  $\tilde{p}_\theta(z|x)$  respectively. For theoretical underpinning, we should also understand this perturbation in terms of objective function and estimating equation.

(1) **Objective function.** In terms of objective function, the MLE loss function is  $D(\theta) = \text{KL}(p_{\text{data}} \| p_\theta)$ . At iteration  $t$ , with fixed  $\theta_t = (\alpha_t, \beta_t)$ , we perturb  $D(\theta)$  to  $S(\theta)$ :

$$S(\theta) = \text{KL}(p_{\text{data}} \| p_\theta) + \text{KL}(\tilde{p}_{\theta_t}(z|x) \| p_\theta(z|x)) - \text{KL}(\tilde{p}_{\alpha_t}(z) \| p_\alpha(z)), \quad (12.79)$$

where

$$\text{KL}(\tilde{p}_{\theta_t}(z|x) \| p_\theta(z|x)) = \mathcal{E}_{p_{\text{data}}(x)} \mathcal{E}_{\tilde{p}_{\theta_t}(z|x)} \left[ \log \frac{\tilde{p}_{\theta_t}(z|x)}{p_\theta(z|x)} \right], \quad (12.80)$$

i.e., the KL-divergence between conditional distributions of  $z$  given  $x$  also integrates over the marginal distribution  $x$  as defined before.

The learning algorithm based on short-run MCMC is  $\theta_{t+1} = \theta_t - \eta_t S'(\theta_t)$ , because

$$S'(\theta_t) = \mathcal{E}_{p_{\text{data}}}[\tilde{\delta}_{\theta_t}(x)]. \quad (12.81)$$

Thus the updating rule of the learning algorithm follows the stochastic gradient (i.e., Monte Carlo approximation of the gradient) of a perturbation of the log-likelihood.

$S(\theta)$  is in the form of divergence triangle, which consists of two perturbations.  $\text{KL}(\tilde{p}_{\theta_t}(z|x)||p_{\theta}(z|x))$  is for inference, and  $\text{KL}(\tilde{p}_{\alpha_t}(z)||p_{\alpha}(z))$  is for synthesis. It is a combination of the perturbations for the descriptive model and generative model respectively.

(2) **Estimating equation.** In terms of estimating equation, the stochastic gradient descent learning is a Robbins-Monro stochastic approximation algorithm that solves the following estimating equation:

$$\frac{1}{n} \sum_{i=1}^n \tilde{\delta}_{\alpha}(x_i) = \frac{1}{n} \sum_{i=1}^n \mathcal{E}_{\tilde{p}_{\theta}(z_i|x_i)}[\nabla_{\alpha} f_{\alpha}(z_i)] - \mathcal{E}_{\tilde{p}_{\alpha}(z)}[\nabla_{\alpha} f_{\alpha}(z)] = 0, \quad (12.82)$$

$$\frac{1}{n} \sum_{i=1}^n \tilde{\delta}_{\beta}(x_i) = \frac{1}{n} \sum_{i=1}^n \mathcal{E}_{\tilde{p}_{\theta}(z_i|x_i)}[\nabla_{\beta} \log p_{\beta}(x_i|z_i)] = 0. \quad (12.83)$$

The solution to the above estimating equation defines an estimator of the parameters. The learning algorithm converges to this estimator under the usual regularity conditions of Robbins-Monro. If we replace  $\tilde{p}_{\alpha}(z)$  by  $p_{\alpha}(z)$ , and  $\tilde{p}_{\theta}(z|x)$  by  $p_{\theta}(z|x)$ , then the above estimating equation is the maximum likelihood estimating equation.

The above theoretical understanding in terms of objective function and estimating equation is more general than that of maximum likelihood, which is a special case where the number of steps  $K \rightarrow \infty$  and the step size  $s \rightarrow 0$  in the Langevin dynamics in equation (12.76). Our theoretical understanding is clearly more relevant in practice where we can only afford finite  $K$  with non-zero  $s$ .

As to the step size  $s$ , we currently treat it as a tuning parameter.  $s$  can be more formally optimized by variational optimization, i.e., by minimizing the KL-divergences that perturb  $D(\theta)$ .

## 12.6 Variational and Adversarial Learning

### 12.6.1 From Short-run MCMC to Learned Sampling Computations

In both descriptive model and generative model, we use short-run MCMC [167, 168] for the sampling computations of synthesis and inference. In this section, we shall study learning methods that replace short-run MCMC by learned computations for synthesis and inference sampling.

One popular learning method is variational auto-encoder (VAE) [121] for the generator model, where the short-run MCMC for inference is replaced by an inference model. The other popular learning method is generative adversarial networks (GAN) [74, 182], which is related to the descriptive model, where we replace short-run MCMC for synthesis by a generator model.

Both learning methods can be theoretically understood by the divergence triangle framework.

### 12.6.2 VAE: learned computation for inference sampling

For the generator model  $p_{\theta}(x, z) = p(z)p_{\theta}(x|z)$  studied in the previous sections, the VAE [121] approximates the posterior  $p_{\theta}(z|x)$  by a tractable  $q_{\phi}(z|x)$ , such as

$$q_{\phi}(z|x) \sim \text{N}(\mu_{\phi}(x), \text{diag}(v_{\phi}(x))), \quad (12.84)$$

where both  $\mu_\phi$  and  $v_\phi$  are bottom-up networks that map  $x$  to  $d$ -dimensional vectors, with  $\phi$  collecting all the weight and bias parameters of the bottom-up networks. For  $z \sim q_\phi(z|x)$ , we can write  $h = \mu_\phi(x) + \text{diag}(v_\phi(x))^{1/2}z$ , where  $z \sim N(0, I_d)$ . Thus expectation with respect to  $z \sim q_\phi(z|x)$  can be written as expectation with respect to  $z$ . This reparameterization trick [121] helps reduce the variance in Monte Carlo integration. We may consider  $q_\phi(z|x)$  as an approximation to the iterative MCMC sampling of  $p_\theta(z|x)$ . In other words,  $q_\phi(z|x)$  is the learned inferential computation that approximately samples from  $p_\theta(z|x)$ .

The VAE objective is a modification of the maximum likelihood estimation (MLE) objective  $D(\theta) = \text{KL}(p_{\text{data}}(x) \| p_\theta(x))$ :

$$S(\theta, \phi) = D(\theta) + \text{KL}(q_\phi(z|x) \| p_\theta(z|x)) \quad (12.85)$$

$$= \text{KL}(p_{\text{data}}(x) \| p_\theta(x)) + \text{KL}(q_\phi(z|x) \| p_\theta(z|x)) \quad (12.86)$$

$$= \text{KL}(p_{\text{data}}(x) q_\phi(z|x) \| p_\theta(z, x)). \quad (12.87)$$

We define the conditional KL divergence

$$\text{KL}(q_\phi(z|x) \| p_\theta(z|x)) = \mathcal{E}_{p_{\text{data}}} \mathcal{E}_{q_\phi(z|x)} \left[ \log \frac{q_\phi(z|x)}{p_\theta(z|x)} \right], \quad (12.88)$$

where we also average over  $p_{\text{data}}$ .

We estimate  $\theta$  and  $\phi$  jointly by

$$\min_{\theta} \min_{\phi} S(\theta, \phi), \quad (12.89)$$

which can be accomplished by gradient descent.

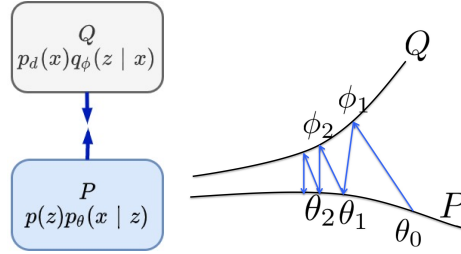


Figure 12.12: Reprinted with permission from [86]. Variational auto-encoder as joint minimization by alternating projection.  $P = p(z)p_\theta(x|z)$  is the distribution of the complete-data model, where  $p(z)$  is the prior distribution of hidden vector  $z$  and  $p_\theta(x|z)$  is the conditional distribution of  $x$  given  $z$ .  $Q = p_d(x)q_\phi(z|x)$  is the distribution of the complete data  $(z, x)$ , where  $p_d(x)$  refers to the data distribution  $p_{\text{data}}$  and  $q_\phi(z|x)$  is the learned inferential computation that approximately samples from the posterior distribution  $p_\theta(x|z)$ . (Left) Interaction between the models. (Right) Alternating projection. The two models run toward each other.

Define  $Q(z, x) = p_{\text{data}}(x)q_\phi(z|x)$ . Define  $P(z, x) = p(z)p_\theta(x|z)$ .  $Q$  is the distribution of the complete data  $(z, x)$ , where  $q_\phi(z|x)$  can be interpreted as an imputer that imputes the missing data  $z$ .  $P$  is the distribution of the complete-data model. Then  $S(\theta, \phi) = \text{KL}(Q \| P)$ . The VAE is  $\min_{\theta} \min_{\phi} \text{KL}(Q \| P)$ .

We may interpret the VAE as an alternating projection between  $Q$  and  $P$ . Figure 12.12 provides an illustration. The wake-sleep algorithm [92] is similar to the VAE, except that it updates  $\phi$  by  $\min_{\phi} \text{KL}(P \| Q)$ , where the order is flipped.

In the VAE, the model  $q_\phi(z|x)$  and the parameter  $\phi$  are shared by all the training examples  $x$ , so that  $\mu_\phi(x)$  and  $v_\phi(x)$  in Equation 12.84 can be computed directly for each  $x$  given  $\phi$ . This is different from

traditional variational inference [18, 112], where for each  $x$ , a model  $q_{\mu,v}(z)$  is learned by minimizing  $\text{KL}(q_{\mu,v}(z)||p_\theta(z|x))$  with  $x$  fixed, so that  $(\mu, v)$  is computed by an iterative algorithm for each  $x$ , which is an inner loop of the learning algorithm. This is similar to maximum likelihood learning, except that in maximum likelihood learning, the inner loop is an iterative algorithm that samples  $p_\theta(z|x)$  instead of minimizing over  $(\mu, v)$ . The learned networks  $\mu_\phi(x)$  and  $v_\phi(d)$  in the VAE are to approximate the iterative minimization algorithm by direct mappings.

### 12.6.3 GAN: joint learning of generator and discriminator

The generator model learned by MLE or the VAE usually cannot generate very realistic images. Both MLE and the VAE target  $\text{KL}(p_{\text{data}}||p_\theta)$ , though the VAE only minimizes an upper bound of  $\text{KL}(p_{\text{data}}||p_\theta)$ . Consider minimizing  $\text{KL}(q||p)$  over  $p$  within a certain model class. If  $q$  is multi-modal, then  $p$  is obliged to fit all the major modes of  $q$  because  $\text{KL}(q||p)$  is an expectation with respect to  $q$ . Thus,  $p$  tends to interpolate the major modes of  $q$  if  $p$  cannot fit the modes of  $q$  closely. As a result,  $p_\theta$  learned by MLE or the VAE tends to generate images that are not as sharp as the observed images.

The behavior of minimizing  $\text{KL}(q||p)$  over  $p$  is different from minimizing  $\text{KL}(q||p)$  over  $q$ . If  $p$  is multi-modal,  $q$  tends to capture some major modes of  $p$  while ignoring the other modes of  $p$ , because  $\text{KL}(q||p)$  is an expectation with respect to  $q$ . In other words,  $\min_q \text{KL}(q||p)$  encourages mode chasing, whereas  $\min_p \text{KL}(q||p)$  encourages mode covering.

Sharp synthesis can be achieved by GAN [74, 182], which pairs a generator model  $G$  with a discriminator model  $D$ . For an image  $x$ ,  $D(x)$  is the probability that  $x$  is an observed (real) image instead of a generated (faked) image. It can be parameterized by a bottom-up network  $f_\alpha(x)$ , so that  $D(x) = 1/(1 + \exp(-f_\alpha(x)))$ , i.e., logistic regression. We can train the pair of  $(G, D)$  by an adversarial, zero-sum game. Specifically, let  $G(z) = g_\theta(z)$  be a generator. Let

$$V(D, G) = \mathcal{E}_{p_{\text{data}}}[\log D(x)] + \mathcal{E}_{z \sim p(z)}[\log(1 - D(G(z)))], \quad (12.90)$$

where  $\mathcal{E}_{p_{\text{data}}}$  can be approximated by averaging over the observed examples, and  $\mathcal{E}_z$  can be approximated by Monte Carlo average over the faked examples generated by the generator model. We learn  $D$  and  $G$  by  $\min_G \max_D V(D, G)$ .  $V(D, G)$  is the log-likelihood for  $D$ , i.e., the log-probability of the real and faked examples. However,  $V(D, G)$  is not a very convincing objective for  $G$ . In practice, the training of  $G$  is usually modified into maximizing  $\mathcal{E}_{z \sim p(z)}[\log D(G(z))]$  to avoid the vanishing gradient problem.

For a given  $\theta$ , let  $p_\theta$  be the distribution of  $g_\theta(z)$  with  $z \sim p(z)$ . Assuming a perfect discriminator. Then, according to the Bayes' theorem,  $D(x) = p_{\text{data}}(x)/(p_{\text{data}}(x) + p_\theta(x))$  (assuming equal numbers of real and faked examples). Then  $\theta$  minimizes the Jensen-Shannon (JS) divergence

$$\text{JS}(p_{\text{data}}||p_\theta) = \text{KL}(p_\theta||p_{\text{mix}}) + \text{KL}(p_{\text{data}}||p_{\text{mix}}), \quad (12.91)$$

where  $p_{\text{mix}} = (p_{\text{data}} + p_\theta)/2$ .

In JS-divergence, the model  $p_\theta$  also appears on the left-hand side of KL divergence. This encourages  $p_\theta$  to fit some major modes of  $p_{\text{data}}$  while ignoring others. As a result, GAN learning suffers from the mode collapsing problem, i.e., the learned  $p_\theta$  may miss some modes of  $p_{\text{data}}$ . However, the  $p_\theta$  learned by GAN tends to generate sharper images than the  $p_\theta$  learned by MLE or the VAE.

### 12.6.4 Joint Learning of Descriptive and Generative Models

We can also learn the descriptive model and the generative model jointly, similar to GAN. In this joint learning scheme, we seek to learn the descriptive model by MLE, following the analysis by synthesis scheme.

But we recruit the generator model as an approximate sampler, i.e., in this context, the generator model is the learned computation for synthesis sampling.

We continue to use  $p_\theta(x, z) = p(z)p_\theta(x|z)$  to denote the generative model, and we denote the descriptive model by

$$\pi_\alpha(x) = \frac{1}{Z(\alpha)} \exp(f_\alpha(x)), \quad (12.92)$$

so that we will not confuse the notation.

To avoid MCMC sampling of  $\pi_\alpha$ , we may approximate it by a generator model  $p_\theta$ , which can generate synthesized examples directly (i.e., sampling  $z$  from  $p(z)$ , and transforming  $z$  to  $x$  by  $x = g_\theta(z)$ ). We may consider  $p_\theta$  an approximation to the iterative MCMC sampling of  $\pi_\alpha$ . In other words,  $p_\theta$  is the learned computation that approximately samples from  $\pi_\alpha$ , it is an approximate direct sampler of  $\pi_\alpha$ .

The MLE learning objective is  $D(\alpha) = \text{KL}(p_{\text{data}} \| \pi_\alpha)$ . We can learn both  $\pi_\alpha$  and  $p_\theta$  using the following objective function [32, 118]:

$$S(\alpha, \theta) = D(\alpha) - \text{KL}(p_\theta \| \pi_\alpha) = \text{KL}(p_{\text{data}} \| \pi_\alpha) - \text{KL}(p_\theta \| \pi_\alpha). \quad (12.93)$$

We learn  $\alpha$  and  $\theta$  by

$$\min_{\alpha} \max_{\theta} S(\alpha, \theta), \quad (12.94)$$

which defines a mini-max game.

The gradient for updating  $\alpha$  becomes

$$\nabla_{\alpha} S(\alpha, \theta) = \nabla_{\alpha} [\mathcal{E}_{p_{\text{data}}}(f_{\alpha}(x)) - \mathcal{E}_{p_{\theta}}(f_{\alpha}(x))], \quad (12.95)$$

where the intractable  $\log Z(\alpha)$  term is canceled.

Because of the negative sign in front of the second KL divergence in Equation 12.93, we need  $\max_{\theta}$  in Equation 12.94, so that the learning becomes adversarial (illustrated in Figure 12.13). Inspired by [91], [86] called Equation 12.94 the adversarial contrastive divergence (ACD). It underlies the work of [32, 118].

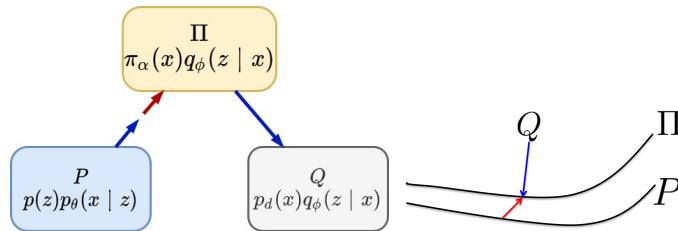


Figure 12.13: Reprinted with permission from [86]. Adversarial contrastive divergence where the energy-based model favors real data against the generator. (Left) Interaction between the models. Red arrow indicates a chasing game, where the red arrow pointing to  $\Pi$  indicates that  $\Pi$  seeks to move away from  $P$ . The blue arrow pointing from  $P$  to  $\Pi$  indicates that  $P$  seeks to move close to  $\Pi$ . (Right) Contrastive divergence.

The adversarial form (Equation 12.94 or 12.93) defines a chasing game with the following dynamics: The generator  $p_\theta$  chases the energy-based model  $\pi_\alpha$  in  $\min_{\theta} \text{KL}(p_\theta \| \pi_\alpha)$ , while the energy-based model  $\pi_\alpha$  seeks to get closer to  $p_{\text{data}}$  and away from  $p_\theta$ . The red arrow in Figure 12.13 illustrates this chasing game.

The result is that  $\pi_\alpha$  lures  $p_\theta$  toward  $p_{\text{data}}$ . In the idealized case,  $p_\theta$  always catches up with  $\pi_\alpha$ , and then  $\pi_\alpha$  will converge to the maximum likelihood estimate  $\min_\alpha \text{KL}(p_{\text{data}}\|\pi_\alpha)$ , and  $p_\theta$  converges to  $\pi_\alpha$ .

This chasing game is different from the VAE  $\min_\theta \min_\phi \text{KL}(Q\|P)$ , which defines a cooperative game where  $q_\phi$  and  $p_\theta$  run toward each other.

Even though the above chasing game is adversarial, both models are running toward the data distribution. While the generator model runs after the energy-based model, the energy-based model runs toward the data distribution. As a consequence, the energy-based model guides or leads the generator model toward the data distribution. It is different from GAN [74], in which the discriminator eventually becomes confused because the generated data become similar to the real data. In the above chasing game, the energy-based model becomes close to the data distribution.

The updating of  $\alpha$  by Equation 12.95 is similar to Wasserstein GAN (WGAN) [7], but unlike WGAN,  $f_\alpha$  defines a probability distribution  $\pi_\alpha$ , and the learning of  $\theta$  is based on  $\min_\theta \text{KL}(p_\theta\|\pi_\alpha)$ , which is a variational approximation to  $\pi_\alpha$ . This variational approximation only requires knowing  $f_\alpha(x)$ , without knowing  $Z(\alpha)$ . However, unlike  $q_\phi(z|x)$ ,  $p_\theta(x)$  is still intractable, in particular, its entropy does not have a closed form. Thus, we can again use variational approximation, by changing the problem  $\min_\theta \text{KL}(p_\theta\|\pi_\alpha)$  to

$$\min_\theta \min_\phi \text{KL}(p(z)p_\theta(x|z)\|\pi_\alpha(x)q_\phi(z|x)). \quad (12.96)$$

Define  $\Pi(z, x) = \pi_\alpha(x)q_\phi(z|x)$ , and then the problem is  $\min_\theta \min_\phi \text{KL}(P\|\Pi)$ , which is analytically tractable and underlies the work of [32]. In fact,

$$\text{KL}(P\|\Pi) = \text{KL}(p_\theta(x)\|\pi_\alpha(x)) + \text{KL}(p_\theta(z|x)\|q_\phi(z|x)). \quad (12.97)$$

Thus, we can use  $\max_\alpha \min_\theta \min_\phi [\text{KL}(P\|\Pi) - \text{KL}(Q\|\Pi)]$ , because  $\text{KL}(Q\|\Pi) = \text{KL}(p_{\text{data}}\|\pi_\alpha)$ .

Note that in the VAE (Equation 12.87), the objective function is in the form of  $\text{KL} + \text{KL}$ , whereas in ACD (Equation 12.93), it is in the form of  $\text{KL} - \text{KL}$ . In both Equation 12.87 and 12.93, the first  $\text{KL}$  is about maximum likelihood. The  $\text{KL} + \text{KL}$  form of the VAE makes the computation tractable by changing the marginal distribution of  $x$  to the joint distribution of  $(z, x)$ . The  $\text{KL} - \text{KL}$  form of ACD makes the computation tractable by cancelling the intractable  $\log Z(\alpha)$  term. Because of the negative sign in Equation 12.93, the ACD objective function becomes an adversarial one or a minimax game.

Also note that in the VAE,  $p_\theta$  appears on the right-hand side of  $\text{KL}$ , whereas in ACD,  $p_\theta$  appears on the left-hand side of  $\text{KL}$ . Thus in ACD,  $p_\theta$  may exhibit mode chasing behavior, i.e., fitting the major modes of  $\pi_\alpha$ , while ignoring other modes.

### 12.6.5 Divergence Triangle: Integrating VAE and ACD

We can combine the VAE and ACD into a divergence triangle, which involves the following three joint distributions on  $(z, x)$  defined above:

1.  $Q$  distribution:  $Q(z, x) = p_{\text{data}}(x)q_\phi(z|x)$
2.  $P$  distribution:  $P(z, x) = p(z)p_\theta(x|z)$
3.  $\Pi$  distribution:  $\Pi(z, x) = \pi_\alpha(x)q_\phi(z|x)$

[86] proposed to learn the three models  $p_\theta$ ,  $\pi_\alpha$ , and  $q_\phi$  by the following divergence triangle loss functional  $S$

$$\max_\alpha \min_\theta \min_\phi S(\alpha, \theta, \phi),$$

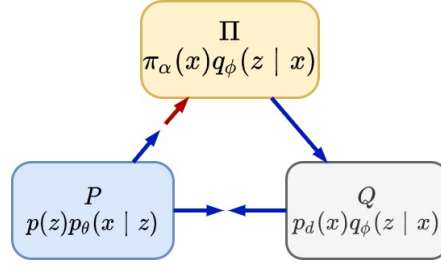


Figure 12.14: Reprinted with permission from [86]. Divergence triangle is based on the Kullback-Leibler divergences between three joint distributions,  $Q$ ,  $P$ , and  $\Pi$ , of  $(z, x)$ . The blue arrow indicates the “running toward” behavior and the red arrow indicates the “running away” behavior.

$$S = \text{KL}(Q\|P) + \text{KL}(P\|\Pi) - \text{KL}(Q\|\Pi). \quad (12.98)$$

See Figure 12.14 for an illustration. The divergence triangle is based on the three KL divergences between the three joint distributions on  $(z, x)$ . It has a symmetric and anti-symmetric form, where the anti-symmetry is due to the negative sign in front of the last KL divergence and the maximization over  $\alpha$ . Compared to the VAE and ACD objective functions in the previous subsections,  $\text{KL}(Q\|P)$  is the VAE part, and  $\text{KL}(P\|\Pi) - \text{KL}(Q\|\Pi)$  is the ACD part.

The divergence triangle leads to the following dynamics between the three models: (a)  $Q$  and  $P$  seek to get close to each other. (b)  $P$  seeks to get close to  $\Pi$ . (c)  $\pi$  seeks to get close to  $p_{\text{data}}$ , but it seeks to get away from  $P$ , as indicated by the red arrow. Note that  $\text{KL}(Q\|\Pi) = \text{KL}(p_{\text{data}}\|\pi_\alpha)$ , because  $q_\phi(z|x)$  is canceled out. The effect of (b) and (c) is that  $\pi$  gets close to  $p_{\text{data}}$  while inducing  $P$  to get close to  $p_{\text{data}}$  as well, or in other words,  $P$  chases  $\pi_\alpha$  toward  $p_{\text{data}}$ .

[86] also employed the layer-wise training scheme of [115] to learn models by divergence triangle from the CelebA-HQ data set [115], including 30,000 celebrity face images with resolutions of up to  $1,024 \times 1,024$  pixels. The learning algorithm converges stably, without extra tricks, to obtain realistic results as shown in Figure 12.15.

Figure 12.15a displays a few  $1,024 \times 1,024$  images generated by the learned generator model with 512-dimensional latent vector. Figure 12.15b shows an example of interpolation. The two images at the two ends are generated by two different latent vectors. The images in between are generated by the vectors that are linear interpolations of the two vectors at the two ends. Even though the interpolation is linear in the latent vector space, the nonlinear mapping leads to a highly nonlinear interpolation in the image space. We first do linear interpolation between the latent vectors at the two ends, i.e.,  $(1 - \alpha) \times z_0 + \alpha \times z_1$ , where  $z_0$  and  $z_1$  are two latent vectors at two ends, respectively, and  $\alpha$  is in the closed unit interval  $[0, 1]$ . The images in between are generated by mapping those interpolated vectors to image space via the learned generator. The interpolation experiment shows that the algorithm can learn a smooth generator model that traces the manifold of the data distribution.

## 12.7 Cooperative Learning via MCMC teaching

### 12.7.1 Joint Training of Descriptive and Generative Models

In ACD, the generator model  $p_\theta$  is used to approximate the energy-based model  $\pi_\alpha$ , and we treat the examples generated by  $p_\theta$  as if they are generated from  $\pi_\alpha$  for the sake of updating  $\alpha$ . The gap between  $p_\theta$  and  $\pi_\alpha$  can cause bias in learning. In the work of [240, 241], we proposed to bring back MCMC to bridge the



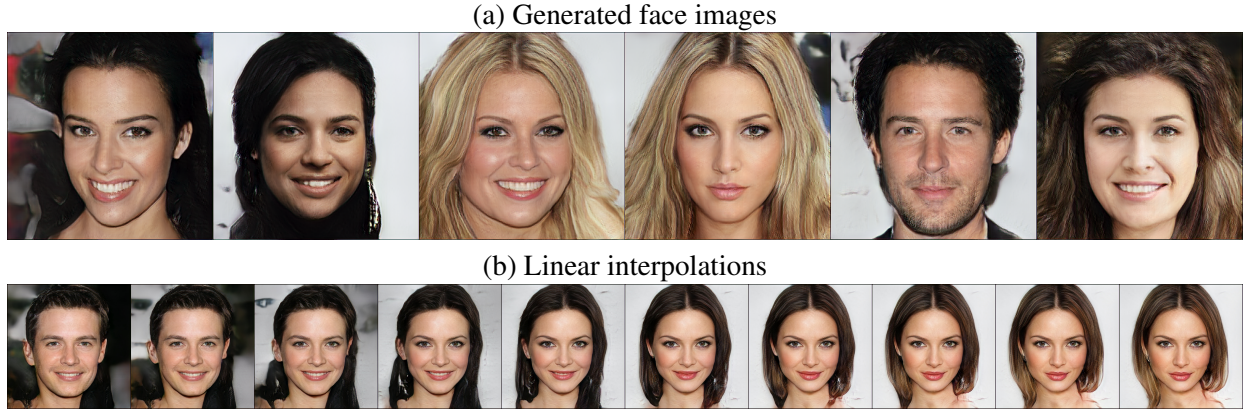
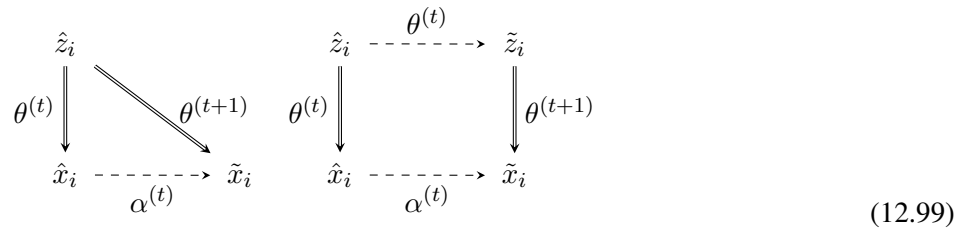


Figure 12.15: Reprinted with permission from [86]. Learning generator model by divergence triangle from the CelebA-HQ data set [115] that includes 30,000 high-resolution celebrity face images. (a) Generated face images with  $1,024 \times 1,024$  resolution sampled from the learned generator model with 512-dimensional latent vector. (b) Linear interpolation of the vector representations. The images at the two ends are generated from latent vectors randomly sampled from a Gaussian distribution. Each image in the middle is obtained by first interpolating the two vectors of the two end images, and then generating the image using the generator.

gap. Instead of running MCMC from scratch, we run a finite-step MCMC toward  $\pi_\alpha$ , initialized from the examples generated by  $p_\theta$ . We then use the examples produced by the finite-step MCMC as the synthesized examples from  $\pi_\alpha$  for updating  $\alpha$ . Meanwhile we update  $p_\theta$  based on how the finite-step MCMC revises the initial examples generated by  $p_\theta$ ; in other words, the energy-based model (as a teacher)  $\pi_\alpha$  distills the MCMC into the generator (as a student)  $p_\theta$ . We call this scheme cooperative learning.

Specifically, we first generate  $\hat{z}_i \sim N(0, I_d)$ , and then generate  $\hat{x}_i = g_\theta(\hat{z}_i) + \epsilon_i$ , for  $i = 1, \dots, \tilde{n}$ . Starting from  $\{\hat{x}_i, i = 1, \dots, \tilde{n}\}$ , we run MCMC such as Langevin dynamics for a finite number of steps toward  $\pi_\alpha$  to get  $\{\tilde{x}_i, i = 1, \dots, \tilde{n}\}$ , which are revised versions of  $\{\hat{x}_i\}$ .  $\{\tilde{x}_i\}$  are used as the synthesized examples from the descriptive model.

The descriptive model can teach the generator via MCMC. The key is that in the generated examples, the latent  $z$  is known. In order to update  $\theta$  of the generator model, we treat  $\{\tilde{x}_i, i = 1, \dots, \tilde{n}\}$  as the training data for the generator. Since these  $\{\tilde{x}_i\}$  are obtained by the Langevin dynamics initialized from  $\{\hat{x}_i\}$ , which are generated by the generator model with known latent factors  $\{\hat{z}_i\}$ , we can update  $\theta$  by learning from the complete data  $\{(\hat{z}_i, \tilde{x}_i); i = 1, \dots, \tilde{n}\}$ , which is a supervised learning problem, or more specifically, a nonlinear regression of  $\tilde{x}_i$  on  $\hat{z}_i$ . At  $\theta^{(t)}$ , the latent factors  $\hat{z}_i$  generates and thus reconstructs the initial example  $\hat{x}_i$ . After updating  $\theta$ , we want  $\hat{z}_i$  to reconstruct the revised example  $\tilde{x}_i$ . That is, we revise  $\theta$  to absorb the MCMC transition from  $\hat{x}_i$  to  $\tilde{x}_i$ . The left panel of diagram (12.99) illustrates the basic idea.



In the two diagrams in (12.99), the double-line arrows indicate generation and reconstruction by the generator model, while the dashed-line arrows indicate Langevin dynamics for MCMC sampling and inference



in the two models. The right panel of diagram (12.99) illustrates a more rigorous method, where we initialize the MCMC for inferring  $\{\tilde{z}_i\}$  from the known  $\{\hat{z}_i\}$  and then update  $\theta$  based on  $\{(\tilde{z}_i, \tilde{x}_i), i = 1, \dots, \tilde{n}\}$ .

The theoretical understanding of the cooperative learning scheme is given below.

(1) Modified contrastive divergence for the energy-based model. In the traditional contrastive divergence [91],  $\hat{x}_i$  is taken to be the observed  $x_i$ . In cooperative learning,  $\hat{x}_i$  is generated by  $p_{\theta(t)}$ . Let  $M_\alpha$  be the Markov transition kernel of finite steps of Langevin dynamics that samples  $\pi_\alpha$ . Let  $(M_\alpha p_\theta)(x) = \int M_\alpha(x', x) p_\theta(x') dx'$  be the marginal distribution by running  $M_\alpha$  initialized from  $p_\theta$ . Then similar to the traditional contrastive divergence, the learning gradient of the evaluator model  $\alpha$  at iteration  $t$  is the gradient of  $\text{KL}(p_{\text{data}} \parallel \pi_\alpha) - \text{KL}(M_{\alpha(t)} p_{\theta(t)} \parallel \pi_\alpha)$  with respect to  $\alpha$ . In the traditional contrastive divergence,  $p_{\text{data}}$  takes the place of  $p_{\theta(t)}$  in the second KL divergence.

(2) MCMC teaching of the generator model. The learning gradient of the generator  $\theta$  in the right panel of diagram (12.99) is the gradient of  $\text{KL}(M_{\alpha(t)} p_{\theta(t)} \parallel p_\theta)$  with respect to  $\theta$ . Here  $\pi^{(t+1)} = M_{\alpha(t)} p_{\theta(t)}$  takes the place of  $p_{\text{data}}$  as the data to train the generator model. It is much easier to minimize  $\text{KL}(M_{\alpha(t)} p_{\theta(t)} \parallel p_\theta)$  than to minimize  $\text{KL}(p_{\text{data}} \parallel p_\theta)$  because the latent variables are essentially known in the former, so the learning is supervised. The MCMC teaching alternates between Markov transition from  $p_{\theta(t)}$  to  $\pi^{(t+1)}$ , and projection from  $\pi^{(t+1)}$  to  $p_{\theta(t+1)}$ , as illustrated by Figure 12.16.

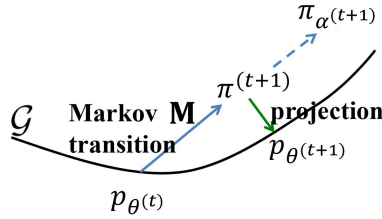


Figure 12.16: Reprinted with permission from [242]. The MCMC teaching of the generator alternates between Markov transition and projection. The family of the generator models  $\mathcal{G}$  is illustrated by the black curve, and each distribution is illustrated by a point.  $p_\theta$  is a generator model, and  $\pi_\alpha$  is a descriptive model.

Figure 12.17 displays two examples of image synthesis by cooperative learning algorithm on datasets, LSUN bedrooms [255] and CelebA human faces [142].

## 12.7.2 Conditional Learning via Fast Thinking Initializer and Slow Thinking Solver

Recently, [245] extended the cooperative learning scheme to the conditional learning problem by jointly learning a conditional energy-based model and a conditional generator model. The conditional energy-based model is of the following form

$$\pi_\alpha(x|c) = \frac{1}{Z(c, \alpha)} \exp[f_\alpha(x, c)], \quad (12.100)$$

where  $x$  is the input signal and  $c$  is the condition.  $Z(c, \alpha)$  is the normalizing constant conditioned on  $c$ .  $f_\alpha(x, c)$  can be defined by a bottom-up ConvNet where  $\alpha$  collects all the weight and bias parameters. Fixing the condition  $c$ ,  $f_\alpha(x, c)$  defines the value of  $x$  for the condition  $c$ , and  $-f_\alpha(x, c)$  defines the conditional energy function.  $\pi_\alpha(x|c)$  is also a deep generalization of conditional random fields [125]. Both the conditional generator model and the conditional energy-based model can be learned jointly by the cooperative learning scheme in Section 12.7.



Figure 12.17: Reprinted with permission from [246]. Image synthesis by cooperative learning. (a) Generating bedroom images ( $256 \times 256$  pixels). The synthesized images are generated by the cooperative learning algorithm that learns from the LSUN data set [255] with 3,033,000 training images. (b) Generating human face images ( $128 \times 128$  pixels). The synthesized images are generated by the cooperative learning algorithm that learns from the CelebA data set [142] with 200,000 training images. For each category, the top panel shows examples of the training images, and the bottom panel shows examples of the synthesized images generated by the learned models.

Figure 12.18 shows some examples of learning the conditional distribution of an image given a class label. The two models are jointly learned on 30,000 handwritten digit images from the MNIST database [127] conditioned on their class labels, which are encoded as one-hot vectors. For each class, 10 randomly sampled images are displayed. Each column is conditioned on one label and each row is a different generated sample.

Figure 12.19 shows some examples of pattern completion on the CMP (Center for Machine Perception) Facades data set [217] by learning a mapping from an occluded image ( $256 \times 256$  pixels), where a mask of the size of  $128 \times 128$  pixels is centrally placed onto the original version, to the original image. In this case,  $c$  is the observed part of the signal, and  $x$  is the unobserved part of the signal.

The cooperative learning of the conditional generator model and conditional energy-based model can be interpreted as follows. The conditional energy function defines the objective function or value function, i.e., it defines what solutions are desirable given the condition or the problem. The solutions can then be obtained by an iterative optimization or sampling algorithm such as MCMC. In other words, the conditional energy-based model leads to a solver in the form of an iterative algorithm, and this iterative algorithm is a slow thinking process. In contrast, the conditional generator model defines a direct mapping from condition or problem to solutions, and it is a fast thinking process. We can use the fast thinking generator as an initializer to generate the initial solution, and then use the slow thinking solver to refine the fast thinking initialization by the iterative algorithm. The cooperative learning scheme enables us to learn both the fast thinking initializer and slow thinking solver. Unlike conditional GAN, the cooperative learning scheme has a slow thinking refining process, which can be important if the fast thinking initializer is not optimal.

In terms of inverse reinforcement learning [1, 264], the conditional energy-based model defines the reward or value function, and the iterative solver defines an optimal control or planning algorithm. The conditional generator model defines a policy. The fast thinking policy is about habitual, reflexive, or impulsive behaviors, while the slow thinking solver is about deliberation and planning. Compared with the policy, the value is usually simpler and more generalizable, because it is in general easier to specify what one wants

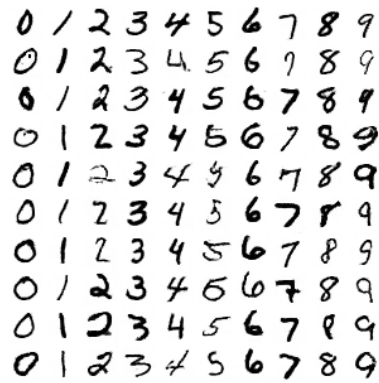


Figure 12.18: Reprinted with permission from [246]. Generated handwritten digits conditioned on class labels. Each column is conditioned on one class label, and each row represents a different generated handwritten digit image. The synthesized images are generated by the jointly trained initializer and solver from 30,000 handwritten digit images along with their class labels from the MNIST database. The image size is  $64 \times 64$  pixels. Abbreviation: MNIST, Modified National Institute of Standards and Technology.

than to specify how to produce what one wants.



Figure 12.19: Reprinted with permission from [246]. Pattern completion by conditional learning. Each row displays one example. The first image is the testing image ( $256 \times 256$  pixels) with a hole of  $128 \times 128$  that needs to be recovered, the second image shows the ground truth, the third image shows the result recovered by the initializer (i.e., conditional generator model), the fourth image shows the result recovered by the solver (i.e., the MCMC sampler of the conditional energy-based model, initialized from the result of the initializer), and the last image shows the result recovered by the conditional GAN as a comparison.

# Bibliography

- [1] Pieter Abbeel and Andrew Y Ng. Apprenticeship learning via inverse reinforcement learning. In *International Conference on Machine Learning*, pages 1–8. ACM, 2004.
- [2] Edward H. Adelson. Layered representations for vision and video. In *In Representation of Visual Scenes, Proceedings IEEE Workshop on International Conference on Computer Vision (ICCV)*, pages 3–9, 1995.
- [3] Adrian Barbu and Song-Chun Zhu. Incorporating visual knowledge representation in stereo reconstruction. In *Tenth IEEE International Conference on Computer Vision (ICCV’05) Volume 1*, volume 1, pages 572–579 Vol. 1, 2005.
- [4] Narendra Ahuja and Mihran Tuceryan. Extraction of early perceptual structure in dot patterns. *CVGIP*, 48, 1989.
- [5] Guillaume Alain and Yoshua Bengio. What regularized auto-encoders learn from the data-generating distribution. *The Journal of Machine Learning Research*, 15(1):3563–3593, 2014.
- [6] Daniel J. Amit. Modeling brain function: The world of attractor neural networks. *Modeling Brain Function*, 1989.
- [7] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan. *arXiv preprint arXiv:1701.07875*, 2017.
- [8] Adrian Barbu and Song-Chun Zhu. Generalizing swendsen-wang to sampling arbitrary posterior probabilities. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 27(8):1239–1253, 2005.
- [9] Horace B Barlow. Possible principles underlying the transformations of sensory messages. 1961.
- [10] David GT Barrett and Benoit Dherin. Implicit gradient regularization. *arXiv preprint arXiv:2009.11162*, 2020.
- [11] Harry G Barrow and Jay M. Tenenbaum. Retrospective on “interpreting line drawings as three-dimensional surfaces”. *Artificial Intelligence*, 59(1-2):71–80, 1993.
- [12] Peter N Belhumeur. A bayesian approach to binocular stereopsis. *International Journal of Computer Vision*, 19(3):237–260, 1996.
- [13] Anthony J Bell and Terrence J Sejnowski. The independent components of natural scenes are edge filters. *Vision Research*, 37(23):3327–3338, 1997.

- [14] Yoshua Bengio, Ian J. Goodfellow, and Aaron Courville. Deep learning. Book in preparation for MIT Press, 2015.
- [15] Yoshua Bengio, Li Yao, Guillaume Alain, and Pascal Vincent. Generalized denoising auto-encoders as generative models. In *Advances in neural information processing systems*, pages 899–907, 2013.
- [16] James R Bergen and EH Adelson. Theories of visual texture perception. *Spatial vision*, 10:114–134, 1991.
- [17] Julian Besag. Spatial interaction and the statistical analysis of lattice systems. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 192–236, 1974.
- [18] David M Blei, Alp Kucukelbir, and Jon D McAuliffe. Variational inference: a review for statisticians. *Journal of the American Statistical Association*, 112(518):859–877, 2017.
- [19] Jeremy S. De Bonet and Paul Viola. A non-parametric multi-scale statistical model for natural images. *Advances in Neural Information Processing*, 10, 1997.
- [20] Leo Breiman, Jerome Friedman, Charles J Stone, and Richard A Olshen. Classification and regression trees. 1984.
- [21] Hilton Bristow, Anders Eriksson, and Simon Lucey. Fast convolutional sparse coding. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 391–398. IEEE, 2013.
- [22] Michael J Brooks and Berthold KP Horn. Shape and source from shading. 1985.
- [23] David Chandler. Introduction to modern statistical mechanics. *Introduction to Modern Statistical Mechanics*, by David Chandler, pp. 288. Foreword by David Chandler. Oxford University Press, Sep 1987. ISBN-10: 0195042778. ISBN-13: 9780195042771, 1, 1987.
- [24] Hong Chen and Song-Chun Zhu. A generative sketch model for human hair analysis and synthesis. *IEEE transactions on pattern analysis and machine intelligence*, 28(7):1025–1040, 2006.
- [25] Scott Shaobing Chen, David L Donoho, and Michael A Saunders. Atomic decomposition by basis pursuit. *SIAM Journal on Scientific Computing*, 20(1):33–61, 1998.
- [26] Sung Nok Chiu, Dietrich Stoyan, Wilfrid S Kendall, and Joseph Mecke. *Stochastic geometry and its applications*. John Wiley & Sons, 2013.
- [27] Timothy F. Cootes, Gareth J. Edwards, and Christopher J. Taylor. Active appearance models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(6):681–685, 2001.
- [28] Thomas M Cover and Joy A Thomas. *Elements of information theory*. John Wiley & Sons, 2012.
- [29] George R Cross and Anil K Jain. Markov random field texture models. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, (1):25–39, 1983.
- [30] Leonardo Da Vinci, Martin Kemp, and Margaret Walker. *Leonardo on painting: anthology of writings by Leonardo da Vinci, with a selection of documents relating to his career as an artist*. Yale Nota Bene, 1989.
- [31] Jifeng Dai, Yang Lu, and Ying Nian Wu. Generative modeling of convolutional neural networks. *arXiv preprint arXiv:1412.6296*, 2014.

- [32] Zihang Dai, Amjad Almahairi, Philip Bachman, Eduard Hovy, and Aaron Courville. Calibrating energy-based generative adversarial networks. In *International Conference on Learning Representations*, 2017.
- [33] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, volume 1, pages 886–893. Ieee, 2005.
- [34] John G Daugman. Uncertainty relation for resolution in space, spatial frequency, and orientation optimized by two-dimensional visual cortical filters. *JOSA A*, 2(7):1160–1169, 1985.
- [35] Stephen Della Pietra, Vincent Della Pietra, and John Lafferty. Inducing features of random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(4):380–393, 1997.
- [36] Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (methodological)*, pages 1–38, 1977.
- [37] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255. IEEE, 2009.
- [38] Emily L Denton, Soumith Chintala, Rob Fergus, et al. Deep generative image models using a laplacian pyramid of adversarial networks. In *NIPS*, pages 1486–1494, 2015.
- [39] Persi Diaconis and David Freedman. On the statistics of vision: the julesz conjecture. *Journal of Mathematical Psychology*, 24(2):112–138, 1981.
- [40] Sven Dickinson and Zygmunt Pizlo. *Shape perception in human and computer vision*. Springer, 2015.
- [41] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real NVP. In *International Conference on Learning Representations*, volume abs/1605.08803, 2017.
- [42] Gianluca Donato and Serge Belongie. Approximate thin plate spline mappings. In *European conference on computer vision*, pages 21–31. Springer, 2002.
- [43] David Leigh Donoho. Sparse components of images and optimal atomic decompositions. *Constructive Approximation*, 17(3):353–382, 2001.
- [44] Alexey Dosovitskiy, Jost Tobias Springenberg, and Thomas Brox. Learning to generate chairs with convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1538–1546, 2015.
- [45] Gareth J Edwards, Christopher J Taylor, and Timothy F Cootes. Learning to identify and track faces in image sequences. In *Proceedings Third IEEE International Conference on Automatic Face and Gesture Recognition*, pages 260–265. IEEE, 1998.
- [46] David Eigen, Christian Puhrsch, and Rob Fergus. Depth map prediction from a single image using a multi-scale deep network. In *Advances in neural information processing systems*, pages 2366–2374, 2014.

- [47] Michael Elad. *Sparse and redundant representations: from theory to applications in signal and image processing*. Springer Science & Business Media, 2010.
- [48] James H. Elder, Amnon Krupnik, and Leigh A. Johnston. Contour grouping with prior models. *Journal of Vision*, 2(4):324–353, 2002.
- [49] James H. Elder, Amnon Krupnik, and Leigh A. Johnston. Contour grouping with prior models. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 25(6):661–674, 2003.
- [50] Selim Esedoglu and Riccardo March. Segmentation with depth but without detecting junctions. *Journal of Mathematical Imaging and Vision*, 18(1):7–15, 2003.
- [51] David J Field. Relations between the statistics of natural images and the response properties of cortical cells. *JOSA A*, 4(12):2379–2394, 1987.
- [52] Chris Fraley and Adrian E Raftery. Model-based clustering, discriminant analysis, and density estimation. *Journal of the American statistical Association*, 97(458):611–631, 2002.
- [53] Robert T. Frankot and Rama Chellappa. A method for enforcing integrability in shape from shading algorithms. *IEEE Transactions on pattern analysis and machine intelligence*, 10(4):439–451, 1988.
- [54] Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139, 1997.
- [55] Brendan J. Frey and Nebojsa Jojic. Transformed component analysis: joint estimation of spatial transforms and image components. In Greece Corfu, editor, *Proc. of 7th ICCV*, 1999.
- [56] A. Fridman. *Mixed Markov Models*. PhD thesis, Division of Applied Math, Brown University, 2000.
- [57] Artur Fridman. Mixed markov field, 2000.
- [58] Artur Fridman. Mixed markov models, applied mathematics. *Proceedings of the National Academy of Sciences*, 100(14):8092–8096, 2003.
- [59] Jerome Friedman, Trevor Hastie, Robert Tibshirani, et al. Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors). *The annals of statistics*, 28(2):337–407, 2000.
- [60] Jerome H Friedman. Exploratory projection pursuit. *Journal of the American Statistical Association*, 82(397):249–266, 1987.
- [61] Jerome H Friedman. Multivariate adaptive regression splines. *The Annals of Statistics*, pages 1–67, 1991.
- [62] Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of Statistics*, 29(5):1189–1232, 2001.
- [63] King-Sun Fu and Bharat K Bhargava. Tree systems for syntactic pattern recognition. *Computers, IEEE Transactions on*, 100(12):1087–1099, 1973.
- [64] Ru-Xin Gao, Tian-Fu Wu, Song-Chun Zhu, and Nong Sang. Bayesian inference for layer representation with mixed markov random field. In *International Workshop on Energy Minimization Methods in Computer Vision and Pattern Recognition*, pages 213–224. Springer, 2007.



- [65] Ruiqi Gao, Yang Lu, Junpei Zhou, Song-Chun Zhu, and Ying Nian Wu. Learning multi-grid generative convnets by minimal contrastive divergence. *arXiv preprint arXiv:1709.08868*, 2017.
- [66] Ruiqi Gao, Yang Lu, Junpei Zhou, Song-Chun Zhu, and Ying Nian Wu. Learning generative ConvNets via multi-grid modeling and sampling. pages 9155–9164, 2018.
- [67] Ruiqi Gao, Yang Song, Ben Poole, Ying Nian Wu, and Diederik P Kingma. Learning energy-based models by diffusion recovery likelihood. *arXiv preprint arXiv:2012.08125*, 2020.
- [68] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013.
- [69] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3354–3361. IEEE, 2012.
- [70] Stuart Geman and Donald Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(4):380–393, 1997.
- [71] Stuart Geman and Christine Graffigne. Markov random field image models and their applications to computer vision. In *Proceedings of the International Congress of Mathematicians*, volume 1, page 2, 1986.
- [72] James J Gibson. The perception of the visual world. 1950.
- [73] Mark Girolami and Ben Calderhead. Riemann manifold langevin and hamiltonian monte carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 73(2):123–214, 2011.
- [74] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.
- [75] Peter J. Green. Reversible jump markov chain monte carlo computation and bayesian model determination. *Biometrika*, 82:711–732, 1995.
- [76] Ulf Grenander. A unified approach to pattern analysis. *Advances in Computers*, 10:175–216, 1970.
- [77] Ulf Grenander and Michael I Miller. *Pattern theory: from representation to inference*. Oxford University Press, 2007.
- [78] Peter Grindrod. *The theory and applications of reaction-diffusion equations: patterns and waves*. Clarendon Press, 1996.
- [79] Ming G. Gu and Fan Hui Kong. A stochastic approximation algorithm with mcmc method for incomplete data estimation problems. *Preprint*, Dept. of Math. and Stat., McGill Univ, 1998.
- [80] Cheng-En Guo, Song-Chun Zhu, and Ying Nian Wu. Modeling visual patterns by integrating descriptive and generative methods. *International Journal of Computer Vision*, 53(1):5–29, 2003.

- [81] Cheng-en Guo, Song-Chun Zhu, and Ying Nian Wu. Towards a mathematical theory of primal sketch and sketchability. In *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pages 1228–1235. IEEE, 2003.
- [82] Cheng-en Guo, Song-Chun Zhu, and Ying Nian Wu. Primal sketch: Integrating structure and texture. *Computer Vision and Image Understanding*, 106(1):5–19, 2007.
- [83] Peter W Hallinan, Gaile Gordon, Alan L Yuille, Peter Giblin, and David Mumford. *Two-and three-dimensional patterns of the face*. AK Peters/CRC Press, 1999.
- [84] Feng Han and Song-Chun Zhu. A two-level generative model for cloth representation and shape from shading. *IEEE Transactions on pattern analysis and machine intelligence*, 29(7):1230–1243, 2007.
- [85] Tian Han, Yang Lu, Song-Chun Zhu, and Ying Nian Wu. Alternating back-propagation for generator network. In *AAAI*, volume 3, page 13, 2017.
- [86] Tian Han, Erik Nijkamp, Xiaolin Fang, Mitch Hill, Song-Chun Zhu, and Ying Nian Wu. Divergence triangle for joint training of generator model, energy-based model, and inference model. *arXiv preprint arXiv:1812.10907*, pages 8670–8679, 2018.
- [87] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning: data mining, inference and prediction*. Springer, 2009.
- [88] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.
- [89] David J Heeger and James R Bergen. Pyramid-based texture analysis/synthesis. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 229–238. ACM, 1995.
- [90] Mitch Hill, Erik Nijkamp, and Song-Chun Zhu. Building a telescope to look into high-dimensional image spaces. *Quarterly of Applied Mathematics*, 77(2):269–321, 2019.
- [91] Geoffrey E Hinton. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14(8):1771–1800, 2002.
- [92] Geoffrey E Hinton, Peter Dayan, Brendan J Frey, and Radford M Neal. The wake-sleep algorithm for unsupervised neural networks, 1995.
- [93] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18(7):1527–1554, 2006.
- [94] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.
- [95] Derek Hoiem, Andrew N Stein, Alexei A Efros, and Martial Hebert. Recovering occlusion boundaries from a single image. In *International Conference on Computer Vision (ICCV)*, pages 1–8, 2007.
- [96] Yi Hong, Zhangzhang Si, Wenze Hu, Song-Chun Zhu, and Ying Nian Wu. Unsupervised learning of compositional sparse code for natural image representation. *Quarterly of Applied Mathematics*, 72:373–406, 2013.

- [97] John J Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, 79(8):2554–2558, 1982.
- [98] Berthold K. P. Horn, Richard S Szeliski, and Alan L. Yuille. Impossible shaded images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(2):166–170, 1993.
- [99] Berthold KP Horn. Shape from shading: A method for obtaining the shape of a smooth opaque object from one view. 1970.
- [100] Berthold KP Horn. The curve of least energy. *CACM Transactions on Mathematical Software*, 9(4):441–460, 1983.
- [101] Berthold KP Horn. Height and gradient from shading. *International journal of computer vision*, 5(1):37–75, 1990.
- [102] David H Hubel and Torsten N Wiesel. Receptive fields, binocular interaction and functional architecture in the cat’s visual cortex. *The Journal of physiology*, 160(1):106, 1962.
- [103] Aapo Hyvärinen. Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 6(Apr):695–709, 2005.
- [104] Aapo Hyvärinen. Connections between score matching, contrastive divergence, and pseudolikelihood for continuous-valued variables. *IEEE Transactions on neural networks*, 18(5):1529–1531, 2007.
- [105] Aapo Hyvärinen, Juha Karhunen, and Erkki Oja. *Independent component analysis*, volume 46. John Wiley & Sons, 2004.
- [106] Katsushi Ikeuchi and Berthold KP Horn. Numerical shape from shading and occluding boundaries. *Artificial intelligence*, 17(1-3):141–184, 1981.
- [107] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.
- [108] Hemant Ishwaran and J Sunil Rao. Spike and slab variable selection: frequentist and bayesian strategies. *The Annals of Statistics*, 33(2):730–773, 2005.
- [109] James R. Bergen and Edward H. Adelson. Early vision and texture perception. *Nature*, 333:363–364, 1988.
- [110] Edwin T Jaynes. Information theory and statistical mechanics. *Physical review*, 106(4):620, 1957.
- [111] Long Jin, Justin Lazarow, and Zhuowen Tu. Introspective classification with convolutional nets. In *Advances in Neural Information Processing Systems*, pages 823–833, 2017.
- [112] Michael I Jordan, Zoubin Ghahramani, Tommi S Jaakkola, and Lawrence K Saul. An introduction to variational methods for graphical models. *Machine Learning*, 37(2):183–233, 1999.
- [113] Bela Julesz. Visual pattern discrimination. *IRE Transactions on Information Theory*, 8(2):84–92, 1962.
- [114] Bela Julesz et al. Textons, the elements of texture perception, and their interactions. *Nature*, 290(5802):91–97, 1981.

- [115] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *International Conference on Learning Representations*, 2017.
- [116] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4401–4410, 2019.
- [117] Hyunsoo Kim and Haesun Park. Nonnegative matrix factorization based on alternating nonnegativity constrained least squares and active set method. *SIAM Journal on Matrix Analysis and Applications*, 30(2):713–730, 2008.
- [118] Taesup Kim and Yoshua Bengio. Deep directed generative models with energy-based probability estimation. *ICLR Workshop*, 2016.
- [119] Benjamin B Kimia, Ilana Frankel, and Ana-Maria Popescu. Euler spiral for shape completion. *International Journal of Computer Vision*, 54(1-3):159–182, 2003.
- [120] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.
- [121] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. *International Conference for Learning Representations*, 2014.
- [122] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.
- [123] Jean Kossaifi, Georgios Tzimiropoulos, and Maja Pantic. Fast and exact newton and bidirectional fitting of active appearance models. *IEEE transactions on image processing*, 26(2):1040–1053, 2017.
- [124] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1097–1105, 2012.
- [125] John Lafferty, Andrew McCallum, and Fernando CN Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *International Conference on Machine Learning*, pages 282–289, 2001.
- [126] Justin Lazarow, Long Jin, and Zhuowen Tu. Introspective neural networks for generative modeling. In *IEEE International Conference on Computer Vision*, pages 2774–2783, 2017.
- [127] Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.
- [128] Yann LeCun, Sumit Chopra, Rata Hadsell, Mare’ Aurelio Ranzato, and Fu Jie Huang. A tutorial on energy-based learning. In *Predicting Structured Data*, volume 1. MIT Press, 2006.
- [129] Ann B Lee, JG Huang, and DB Mumford. Random collage model for natural images. *International Journal of Computer Vision*, 2000.
- [130] Ann B. Lee, David B. Mumford, and Jinggang G. Huang. Occlusion models for natural images: A statistical study of a scale-invariant dead leaves model. *Int’l J. of Computer Vision*, 41(1):35–59, 2001.

- [131] Chia-Hoang Lee and Azriel Rosenfeld. Improved methods of estimating shape from shading using the light source coordinate system. *artificial Intelligence*, 26(2):125–143, 1985.
- [132] Daniel D Lee and H Sebastian Seung. Algorithms for non-negative matrix factorization. In *Advances in Neural Information Processing Systems*, pages 556–562, 2001.
- [133] Kwonjoon Lee, Weijian Xu, Fan Fan, and Zhuowen Tu. Wasserstein introspective neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3702–3711, 2018.
- [134] Thomas Leung and Jitendra Malik. Detecting, localizing and grouping repeated scene elements from an image. In *Proc. 4th ECCV*, , UK, 1996.
- [135] Thomas Leung and Jitendra Malik. Recognizing surface using three-dimensional textons. In Greece Corfu, editor, *Proc. of 7th ICCV*, 1999.
- [136] Michael S. Lewicki and Bruno A. Olshausen. A probabilistic framework for the adaptation and comparison of image codes. *JOSA*, 16(7):1587–1601, 1999.
- [137] Michael H Lin and Carlo Tomasi. Surfaces with occlusions from layered stereo. In *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, volume 1, pages I–I. IEEE, 2003.
- [138] Ce Liu, Song-Chun Zhu, and Heung-Yeung Shum. Learning inhomogeneous gibbs model of faces by minimax entropy. In *International Conference on Computer Vision*, volume 1, pages 281–287. IEEE, 2001.
- [139] Chuanhai Liu, Donald B Rubin, and Ying Nian Wu. Parameter expansion to accelerate em: The px-em algorithm. *Biometrika*, 85(4):755–770, 1998.
- [140] Fayao Liu, Chunhua Shen, and Guosheng Lin. Deep convolutional neural fields for depth estimation from a single image. In *CVPR*, 2015.
- [141] Jun S Liu. *Monte Carlo strategies in scientific computing*. Springer Science & Business Media, 2008.
- [142] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *International Conference on Computer Vision*, pages 3730–3738, December 2015.
- [143] David G Lowe. Object recognition from local scale-invariant features. In *Proceedings of the seventh IEEE international conference on computer vision*, volume 2, pages 1150–1157. Ieee, 1999.
- [144] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.
- [145] Yang Lu, Ruiqi Gao, Song-Chun Zhu, and Ying Nian Wu. Exploring generative perspective of convolutional neural networks by learning random field models. *Statistics and Its Interface*, 11(3):515–529, 2018.
- [146] Yang Lu, Song-Chun Zhu, and Ying Nian Wu. Learning FRAME models using CNN filters. *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.
- [147] J. Malik, S. Belongie, J. Shi, and T. Leung. Textons, contours and regions: Cue integration in image segmentation. In Greece Corfu, editor, *Proc. of 7th ICCV*, 1999.

- [148] J. Malik and P. Perona. Preattentive texture discrimination with early vision mechanisms. *J. of Optical Society of America A*, 7, May 1990.
- [149] Jitendra Malik. Interpreting line drawings of curved objects, 1986.
- [150] S. Mallat and Z. Zhang. Matching pursuit in a time-frequency dictionary. *IEEE trans. on Signal Processing*, 41:3397–3415, 1993.
- [151] Stephane G Mallat. A theory for multiresolution signal decomposition: the wavelet representation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 11(7):674–693, 1989.
- [152] Stéphane G Mallat and Zhifeng Zhang. Matching pursuits with time-frequency dictionaries. *IEEE Transactions on Signal Processing*, 1993.
- [153] David Marr. Vision: a computational investigation into the human representation and processing of visual information. *WH San Francisco: Freeman and Company*, 1(2), 1982.
- [154] David Marr and Herbert Keith Nishihara. Representation and recognition of the spatial organization of three-dimensional shapes. *Proc. R. Soc. Lond. B*, 200(1140):269–294, 1978.
- [155] Georges Matheron. *Random sets and integral geometry*. John Wiley & Sons, 1975.
- [156] Lars Mescheder, Andreas Geiger, and Sebastian Nowozin. Which training methods for gans do actually converge? *arXiv preprint arXiv:1801.04406*, 2018.
- [157] Alfred M. Bruckstein Michal Aharon, Michael Elad. The k-svd: An algorithm for designing of overcomplete dictionaries for sparse representations. *IEEE Transactions On Signal Processing*, 15(12):3736–3745, 2006.
- [158] Ennio Mingolla and James T Todd. Perception of solid shape from shading. *Biological cybernetics*, 53(3):137–151, 1986.
- [159] Andriy Mnih and Karol Gregor. Neural variational inference and learning in belief networks. In *ICML*, pages 1791–1799, 2014.
- [160] Guido F Montufar, Razvan Pascanu, Kyunghyun Cho, and Yoshua Bengio. On the number of linear regions of deep neural networks. In *Advances in Neural Information Processing Systems*, pages 2924–2932, 2014.
- [161] David Mumford. The statistical description of visual signals. *MATHEMATICAL RESEARCH*, 87:233–256, 1996.
- [162] David Mumford and Jayant Shah. Optimal approximations by piecewise smooth functions and associated variational problems. *Communications on pure and applied mathematics*, 42(5):577–685, 1989.
- [163] James D. Murray. A pre-pattern formation mechanism for animal coat markings. *Journal of Theoretical Biology*, 88(1):161–199, 1981.
- [164] Radford M Neal et al. Mcmc using hamiltonian dynamics. *Handbook of markov chain monte carlo*, 2(11):2, 2011.

- [165] Jiquan Ngiam, Zhenghao Chen, Pang W Koh, and Andrew Y Ng. Learning deep energy models. pages 1105–1112, 2011.
- [166] Erik Nijkamp, Mitch Hill, Tian Han, Song-Chun Zhu, and Ying Nian Wu. On the anatomy of mcmc-based maximum likelihood learning of energy-based models. *arXiv*, 2019.
- [167] Erik Nijkamp, Bo Pang, Tian Han, Linqi Zhou, Song-Chun Zhu, and Ying Nian Wu. Learning multi-layer latent variable model via variational optimization of short run mcmc for approximate inference. *European Conference on Computer Vision*, 2020.
- [168] Erik Nijkamp, Song-Chun Zhu, and Ying Nian Wu. Learning non-convergent short-run MCMC toward energy-based model. *NeurIPS*, 2019.
- [169] Mark Nitzberg and David Mumford. The 2.1-d sketch. In *International Conference on Computer Vision (ICCV)*, pages 138–144, 1990.
- [170] Mark Nitzberg and Takahiro Shiota. Nonlinear image filtering with edge and corner enhancement. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (8):826–833, 1992.
- [171] Bruno A Olshausen. Learning sparse, overcomplete representations of time-varying natural images. In *Image Processing, 2003. ICIP 2003. Proceedings. 2003 International Conference on*, volume 1, pages I–41. IEEE, 2003.
- [172] Bruno A. Olshausen and David J. Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381:607–609, 1996.
- [173] Bruno A Olshausen and David J Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381(6583):607, 1996.
- [174] Bruno A Olshausen and David J Field. Sparse coding with an overcomplete basis set: A strategy employed by V1? *Vision Research*, 37(23):3311–3325, 1997.
- [175] Bo Pang, Tian Han, Erik Nijkamp, Song-Chun Zhu, and Ying Nian Wu. Learning latent space energy-based prior model. *arXiv preprint arXiv:2006.08205*, 2020.
- [176] Razvan Pascanu, Guido Montufar, and Yoshua Bengio. On the number of response regions of deep feed forward networks with piece-wise linear activations. *arXiv preprint arXiv:1312.6098*, 2013.
- [177] Alex P Pentland. Local shading analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (2):170–187, 1984.
- [178] Pietro Perona and Jitendra Malik. Scale-space and edge detection using anisotropic diffusion. *IEEE Transactions on pattern analysis and machine intelligence*, 12(7):629–639, 1990.
- [179] Tsai Ping-Sing and Mubarak Shah. Shape from shading using linear approximation. *Image and Vision computing*, 12(8):487–498, 1994.
- [180] Javier Portilla and Eero P. Simoncelli. A parametric texture model based on joint statistics of complex wavelet coefficients. *IJCV*, 40:1, 2000.
- [181] Bruno Poucet and Etienne Save. Attractors in memory. *Science*, 308(5723):799–800, 2005.

- [182] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.
- [183] Xiaofeng Ren, Charless C Fowlkes, and Jitendra Malik. Scale-invariant contour completion using conditional random fields. In *International Conference on Computer Vision (ICCV)*, pages 1214–1221, 2005.
- [184] Xiaofeng Ren, Charless C Fowlkes, and Jitendra Malik. Figure/ground assignment in natural images. In *European Conference on Computer Vision (ECCV)*, pages 614–627, 2006.
- [185] Danilo J. Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In Tony Jebara and Eric P. Xing, editors, *ICML*, pages 1278–1286. JMLR Workshop and Conference Proceedings, 2014.
- [186] Herbert Robbins and Sutton Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951.
- [187] Saharon Rosset, Ji Zhu, and Trevor Hastie. Boosting as a regularized path to a maximum margin classifier. *The Journal of Machine Learning Research*, 5:941–973, 2004.
- [188] Stefan Roth and Michael J Black. Fields of experts: A framework for learning image priors. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 860–867. IEEE, 2005.
- [189] Sam Roweis and Zoubin Ghahramani. A unifying review of linear gaussian models. *Neural Computation*, 11:2, 1999.
- [190] Donald B Rubin. *Multiple imputation for nonresponse in surveys*, volume 81. John Wiley & Sons, 2004.
- [191] Donald B Rubin and Dorothy T Thayer. EM algorithms for ML factor analysis. *Psychometrika*, 47(1):69–76, 1982.
- [192] Daniel L Ruderman. The statistics of natural images. *Network: computation in neural systems*, 5(4):517–548, 1994.
- [193] Eric Saund. Perceptual organization of occluding contours generated by opaque surfaces. In *Computer Vision and Pattern Recognition (CVPR)*, pages 624–630, 1999.
- [194] Eric Saund. Perceptual organization of occluding contours of opaque surfaces. *Computer Vision and Image Understanding*, 76(1):70–82, 1999.
- [195] Daniel Scharstein and Richard Szeliski. High-accuracy stereo depth maps using structured light. In *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, volume 1, pages I–I. IEEE, 2003.
- [196] Stan Sclaroff and John Isidoro. Active blobs. In *Sixth International Conference on Computer Vision (IEEE Cat. No. 98CH36271)*, pages 1146–1153. IEEE, 1998.
- [197] Thomas Serre, Lior Wolf, Stanley Bileschi, Maximilian Riesenhuber, and Tomaso Poggio. Robust object recognition with cortex-like mechanisms. *IEEE transactions on pattern analysis and machine intelligence*, 29(3):411–426, 2007.



- [198] Claude Elwood Shannon. A mathematical theory of communication. *ACM SIGMOBILE Mobile Computing and Communications Review*, 5(1):3–55, 2001.
- [199] Zhangzhang Si and Song-Chun Zhu. Learning hybrid image templates (hit) by information projection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 99(7):1354–1367, 2011.
- [200] Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor segmentation and support inference from rgb-d images. In *European Conference on Computer Vision*, pages 746–760. Springer, 2012.
- [201] Eero P Simoncelli, William T Freeman, Edward H Adelson, and David J Heeger. Shiftable multiscale transforms. *Information Theory, IEEE Transactions on*, 38(2):587–607, 1992.
- [202] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *ICLR*, 2015.
- [203] Steven Skiena. *Combinatorics and graph theory with mathematica*, 1990.
- [204] Samuel L Smith, Benoit Dherin, David GT Barrett, and Soham De. On the origin of implicit regularization in stochastic gradient descent. *arXiv preprint arXiv:2101.12176*, 2021.
- [205] Song-Chun Zhu, Xiuwen Liu, and Ying Nian Wu. Exploring julesz texture ensemble by effective markov chain monte carlo. *PAMI*, 22:6, June 2000.
- [206] Stella X. Yu, Tai Sing Lee, and Takeo Kanade. A hierarchical markov random field model for figure-ground segregation. In *Proc. 4th International Conference on Energy Minimization Methods in Computer Vision and Pattern Recognition (EMMCVPR)*, pages 118–131, 2002.
- [207] Kevin Swersky, Marc’ Aurelio Ranzato, David Buchman, Benjamin Marlin, and Nando Freitas. On autoencoders and score matching for energy based models. In Lise Getoor and Tobias Scheffer, editors, *ICML, ICML ’11*, pages 1201–1208, New York, NY, USA, June 2011. ACM.
- [208] Richard Szeliski. Fast shape from shading. *CVGIP: Image Understanding*, 53(2):129–153, 1991.
- [209] Martin A. Tanner. *Tools for Statistical Inference*. Springer, 1996.
- [210] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, 58(1):267–288, 1996.
- [211] Tijmen Tieleman. Training restricted Boltzmann machines using approximations to the likelihood gradient. In *International Conference on Machine Learning*, pages 1064–1071. ACM, 2008.
- [212] Antonio Torralba and Aude Oliva. Depth estimation from image structure. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 24(9):1226–1238, 2002.
- [213] Zhuowen Tu. Learning generative models via discriminative approaches. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2007.
- [214] Zhuowen Tu and Song-Chun Zhu. Image segmentation by data-driven markov chain monte carlo. *IEEE Transactions on pattern analysis and machine intelligence*, 24(5):657–673, 2002.
- [215] Zhuowen Tu and Song-Chun Zhu. Parsing images into regions, curves and curve groups. *International Journal of Computer Vision*, 69(2):223–249, 2006.

- [216] Alan Mathison Turing. The chemical basis of morphogenesis. *Bulletin of mathematical biology*, 52(1-2):153–197, 1990.
- [217] Radim Tyleček and Radim Šára. Spatial pattern templates for recognition of objects with regular structure. In *German Conference on Pattern Recognition*, pages 364–374. Springer, 2013.
- [218] Omar E Vega and Yee-Hong Yang. Shading logic: A heuristic approach to recover shape from shading. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(6):592–597, 1993.
- [219] Pascal Vincent. A connection between score matching and denoising autoencoders, 2010.
- [220] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *International Conference on Machine Learning*, pages 1096–1103. ACM, 2008.
- [221] David Waltz. Understanding line drawings of scenes with shadows, 1975.
- [222] Jingbin Wang, M Betke, and Erdan Gu. Mosaicshape: Stochastic region grouping with shape prior. In *Computer Vision and Pattern Recognition (CVPR)*, pages 902–908, 2005.
- [223] John YA Wang and Edward H Adelson. Layered representation for motion analysis. In *Computer Vision and Pattern Recognition (CVPR)*, pages 361–366, 1993.
- [224] John YA Wang and Edward H Adelson. Representing moving images with layers. *IEEE Trans. Image Processing*, 3(5):625–638, 1994.
- [225] Yizhou Wang and Song-Chun Zhu. Analysis and synthesis of textured motion: Particles and waves. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(10):1348–1363, 2004.
- [226] Yizhou Wang and Song-Chun Zhu. Perceptual scale-space and its applications. *International Journal of Computer Vision*, 80(1):143–165, 2008.
- [227] Max Welling. Herding dynamical weights to learn. In *International Conference on Machine Learning*, pages 1121–1128. ACM, 2009.
- [228] Max Welling, Richard S Zemel, and Geoffrey E Hinton. Self supervised boosting. In *Advances in Neural Information Processing Systems*, pages 665–672, 2002.
- [229] Lance R. Willianms and Allen R. Hanson. Perceptual completion of occluded surfaces. *Computer Vision and Image Understanding(CVIU)*, 64(1):1–20, 1996.
- [230] Tian-Fu Wu, Gui-Song Xia, and Song-Chun Zhu. Compositional boosting for computing hierarchical image structures. In *IEEE conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, 2007.
- [231] Ying Nian Wu, Ruiqi Gao, Tian Han, and Song-Chun Zhu. A tale of three probabilistic families: Discriminative, descriptive and generative models. *Quarterly of Applied Mathematics*, 77(2):423–465, 2019.
- [232] Ying Nian Wu, Zhangzhang Si, Haifeng Gong, and Song-Chun Zhu. Learning active basis model for object detection and recognitio. *International Journal of Computer Vision*, 90:198–235, 2010.

- [233] Ying Nian Wu, Song-Chun Zhu, and Xiuwen Liu. *Equivalence of Julesz and Gibbs Ensembles*. ICCV, 1999.
- [234] Ying Nian Wu, Jianwen Xie, Yang Lu, and Song-Chun Zhu. Sparse and deep generalizations of the frame model. *Annals of Mathematical Sciences and Applications*, 3(1):211–254, 2018.
- [235] Ying Nian Wu, Song-Chun Zhu, and Cheng-En Guo. From information scaling of natural images to regimes of statistical models. *Quarterly of Applied Mathematics*, 66:81–122, 2008.
- [236] Ying Nian Wu, Song-Chun Zhu, and Xiuwen Liu. Equivalence of julesz and gibbs texture ensembles. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*, volume 2, pages 1025–1032. IEEE, 1999.
- [237] Ying Nian Wu, Song Chun Zhu, and Xiuwen Liu. Equivalence of julesz ensembles and frame models. *International Journal of Computer Vision*, 38(3):247–265, 2000.
- [238] Jianwen Xie, Ruiqi Gao, Zilong Zheng, Song-Chun Zhu, and Ying Nian Wu. Learning dynamic generator model by alternating back-propagation through time. 2019.
- [239] Jianwen Xie, Wenze Hu, Song-Chun Zhu, and Ying Nian Wu. Learning sparse frame models for natural image patterns. *International Journal of Computer Vision*, 114:1–22, 2014.
- [240] Jianwen Xie, Yang Lu, Ruiqi Gao, and Ying Nian Wu. Cooperative learning of energy-based model and latent variable model via MCMC teaching. In *The AAAI Conference on Artificial Intelligence*, 2018.
- [241] Jianwen Xie, Yang Lu, Ruiqi Gao, Song-Chun Zhu, and Ying Nian Wu. Cooperative training of descriptor and generator networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (preprints), 2018.
- [242] Jianwen Xie, Yang Lu, Ruiqi Gao, Song-Chun Zhu, and Ying Nian Wu. Cooperative training of descriptor and generator networks. *IEEE transactions on pattern analysis and machine intelligence*, 42(1):27–45, 2018.
- [243] Jianwen Xie, Yang Lu, Song-Chun Zhu, and Ying Nian Wu. Inducing wavelets into random fields via generative boosting. *Journal of Applied and Computational Harmonic Analysis*, 41:4–25, 2016.
- [244] Jianwen Xie, Yang Lu, Song-Chun Zhu, and Ying Nian Wu. A theory of generative ConvNet. *International Conference on Machine Learning*, pages 2635–2644, 2016.
- [245] Jianwen Xie, Zilong Zheng, Xiaolin Fang, Song-Chun Zhu, and Ying Nian Wu. Multimodal conditional learning with fast thinking policy-like model and slow thinking planner-like model. *arXiv preprint arXiv:1902.02812*, 2019.
- [246] Jianwen Xie, Zilong Zheng, Xiaolin Fang, Song-Chun Zhu, and Ying Nian Wu. Cooperative training of fast thinking initializer and slow thinking solver for conditional learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.
- [247] Jianwen Xie, Zilong Zheng, Ruiqi Gao, Wenguan Wang, Song-Chun Zhu, and Ying Nian Wu. Learning descriptor networks for 3D shape synthesis and analysis. pages 8629–8638, 2018.

- [248] Jianwen Xie, Song-Chun Zhu, and Ying Nian Wu. Synthesizing dynamic patterns by spatial-temporal generative ConvNet. pages 7093–7101, 2017.
- [249] Xianglei Xing, Ruiqi Gao, Tian Han, Song-Chun Zhu, and Ying Nian Wu. Deformable generator networks: Unsupervised disentanglement of appearance and geometry. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–1, 2020.
- [250] Xianglei Xing, Tian Han, Ruiqi Gao, Song-Chun Zhu, and Ying Nian Wu. Unsupervised disentanglement of appearance and geometry by deformable generator network. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10354–10363, June 2019.
- [251] Xianglei Xing, Tianfu Wu, Song-Chun Zhu, and Ying Nian Wu. Inducing hierarchical compositional model by sparsifying generator network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14296–14305, 2020.
- [252] Xianglei Xing, Song-Chun Zhu, and Ying Nian Wu. Inducing sparse coding and and-or grammar from generator network. In *AAAI workshop on Network Interpretability for Deep Learning*, 2019.
- [253] Benjamin Yao, Xiong Yang, and Song-Chun Zhu. An integrated image annotation tool and large scale ground truth database. In *Proc.6th International Conference on Energy Minimization Methods in Computer Vision and Pattern Recognition (EMMCVPR)*, pages 169–183, 2007.
- [254] Laurent Younes. On the convergence of markovian stochastic algorithms with rapidly decreasing ergodicity rates. *Stochastics: An International Journal of Probability and Stochastic Processes*, 65(3-4):177–228, 1999.
- [255] Fisher Yu, Ari Seff, Yinda Zhang, Shuran Song, Thomas Funkhouser, and Jianxiong Xiao. LSUN: Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv preprint arXiv:1506.03365*, abs/1506.03365, 2015.
- [256] Qang Zheng and Rama Chellappa. Estimation of illumination direction, albedo, and shape from shading. *IEEE Trans. on PAMI*, 1991.
- [257] S. C. Zhu. Embedding gestalt laws in markov random fields. *PAMI*, 21:11, 1999.
- [258] Song-Chun Zhu, Cheng-En Guo, Yizhou Wang, and Zijian Xu. What are textons? *International Journal of Computer Vision*, 62(1-2):121–143, 2005.
- [259] Song-Chun Zhu and David Mumford. Grade: Gibbs reaction and diffusion equations. In *ICCV*, pages 847–854. IEEE, 1998.
- [260] Song-Chun Zhu, David Mumford, et al. A stochastic grammar of images. *Foundations and Trends® in Computer Graphics and Vision*, 2(4):259–362, 2007.
- [261] Song-Chun Zhu and David B. Mumford. Prior learning and gibbs reaction-diffusion. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 19(11):1236–1250, 1997.
- [262] Song-Chun Zhu, Ying Nian Wu, and David Mumford. Minimax entropy principle and its application to texture modeling. *Neural Computation*, 9(8):1627–1660, 1997.
- [263] Song-Chun Zhu, Ying Nian Wu, and David Mumford. Filters, random fields and maximum entropy (frame): Toward a unified theory for texture modeling. *International Journal of Computer Vision*, 27(2):107–126, 1998.

- [264] Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, and Anind K Dey. Maximum entropy inverse reinforcement learning. In *The AAAI Conference on Artificial Intelligence*, 2008.