Computer Vision

Statistical Models for Marr's Paradigm

Zhu, Song-Chun Wu, Ying Nian

January 7, 2020

Contents

Authors

Song-Chun Zhu received his Ph.D. degree in Computer Science from Harvard University in 1996 (advised by Dr. David Mumford). He is currently professor of Statistics and Computer Science, and director of the Center for Vision, Learning, Cognition, and Autonomy, at the University of California, Los Angeles. His research interest has been to pursue a common statistical framework for vision, and broadly intelligence. He has published over 300 papers in computer vision, statistical learning, cognition, and AI, and robot autonomy, and received a number of honors, including the David Marr Prize in 2003 for image parsing with Z. Tu et al., the Marr Prize honorary nominations in 1999 for texture modeling and 2007 for object modeling with Dr. Yingnian Wu et al. As a junior faculty he received in 2001 the Sloan Fellow in Computer Science, NSF Career Award, and ONR Young Investigator Award. In 2008 he received the Aggarwal prize from the Intl Association of Pattern Recognition for "contributions to a unified foundation for visual pattern conceptualization, modeling, learning, and inference". In 2013 he received the Helmholtz Test-of-time prize for a paper on image segmentation. He is a fellow of IEEE Computer Society since 2011. He has been the principal investigator leading several ONR MURI and DARPA teams working on scene and event understanding and cognitive robots under a unified mathematical framework.

Ying Nian Wu received his Ph.D. degree in Statistics from Harvard in 1996. He was an assistant professor from 1997 to 1999 in the Department of Statistics, University of Michigan. He joined the University of California, Los Angeles (UCLA) in 1999, and is currently a professor in the Department of Statistics, UCLA. His research interests include statistical modeling, computing, and learning, with applications in computer vision. He has been collaborating with Dr. Song-Chun Zhu in the past 25 years on statistical modeling of images, statistical and machine learning theory. He shared the Marr prize honorary nomination with Dr. Zhu in 1999 and 2007.

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.*

In his monumental book *Vision* (1982) David Marr laid the foundation of modern Computer Vision and influenced a generation of scientists in Neuroscience and Psychology. In *Vision*, Marr proposes a holistic framework for understanding visual perception and touches on broader questions about how cognition can be studied.

Now, thirty years later, the main problems that occupied Marr remain fundamental open problems in the study of perception. This book can be understood as a bridge between Marr's theory of visual perception and the modern treatment of computer vision with statistical modelling. The authors not only explore deep connections between Marr's concepts and observations in Neuroscience but also solidify the findings by a mathematical foundation.

This book presents a unifying mathematical foundation for the study of visual perception and cognition. The proposed foundation integrates different regimes of representations, or models, which cover the fundamental building blocks of visual coding, recognition, cognition, and reasoning. Each building block is systematically presented from a conceptual view, findings in Psychology and Neuroscience, underlying mathematical theory and statistical models, and links to Physics and Statistics. The reader will gain a unified, cross-disciplinary view with facets spanning from Psychology to Statistics rather than being presented a disconnected set of technical machine learning methods.

The authors would like to tribute this book to pioneers in vision - Béla Julesz, David Marr, King-Sun Fu, Ulf Grenander, David Mumford, - and provide foundation, and more importantly, inspiration for the continuing efforts to gain a deeper understanding of visual perception and cognition.

1

Introduction

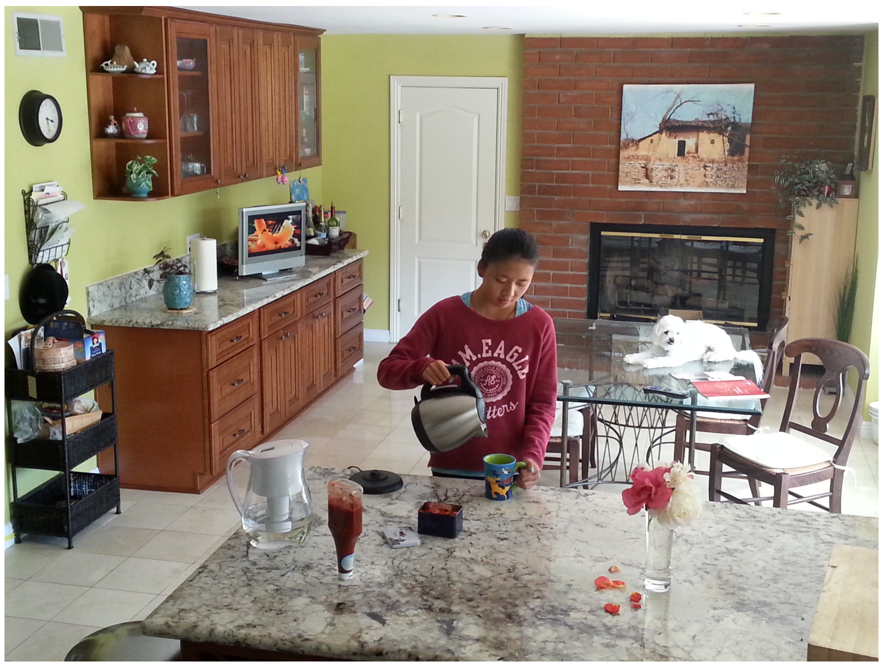


Figure 1.1: A picture is worth a thousand words. Can you come up with a thousand words to describe the above picture?

This introductory chapter gives an overview of the goals that we try to achieve and the methods that we shall adopt in this book.

1.1 Goal of Vision: Understanding Scene and Events

As the saying goes, “a picture is worth a thousand words.” When we look at Fig. 1.1, indeed many words, sentences, or perhaps even a story line may come to mind.

The central theme of this picture is , while a dog is resting on a glass table.

But there is certainly a lot more in the picture. We infer the spatial layout of the room. We recognize many familiar objects. We know their geometric shapes and their positions relative to each other. In addition

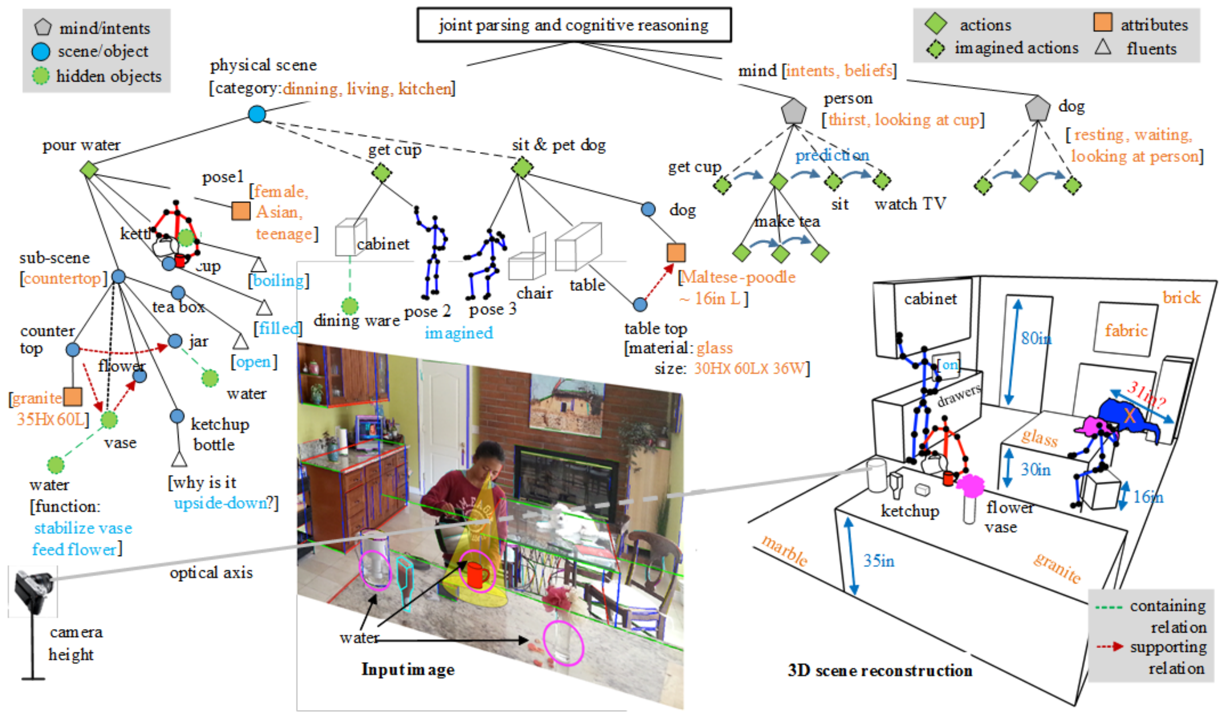


Figure 1.2: Understanding scene and events by joint parsing and cognitive reasoning. From a single image, we aim at reconstructing dense 3D scene, estimating camera parameters, materials and illumination, parsing the scene hierarchically with attributes, fluents and relations, reasoning intents and beliefs of agents (human and dog) and predicting their actions in time, and recovering invisible stuff like water, object states, etc.

to "what is where", we also understand many other concepts. We know the functionalities of the objects in the scene, i.e., what is the purpose of objects such as tables, chairs, cabinet, and how we would use them if we were in such a scene. Back to the central theme, we know that the girl is a teenager. She has just finished the process of making tea. Her current attention is on the cup while she is pouring tea from the teapot. We may infer that she is thirsty. We may also imagine that she may intend to sit down and watch TV while enjoying the cup of tea.

Fig. 1.2 depicts a more formal interpretation of the picture in the form of a parsing graph. There are several types of nodes in this graph, including scene/objects, mind/intents, hidden objects, as well as actions, imagined actions, attributes, and fluents (i.e., how things change over time). These nodes are organized in a tree-like hierarchical structure, with possible connections between sibling nodes. The graph in Fig. 1.2 is an instance of the so-called spatial-temporal-causal And-Or graphs, which we shall elaborate in later chapters. Suffice it to say here, that the goal of vision is to interpret the scene and events in a picture (or a video sequence) in terms of the "words" such as those nodes in Fig. 1.2, as well as their relationships.

Those "words" may be the words in an English dictionary, but they do not have to correspond to any words in a vocal language. Those are "visual words" so to speak. A primate may well possess dictionaries of such "visual words", even though they are not endowed with any capacity for language. We may also call these "visual words" as "symbols" or "patterns." If we want to be more formal, we can call them "concepts," a term that we shall adopt in the discourse of this book, although we shall also use "words," "symbols" and "patterns" at times when appropriate.

A common misconception is that the concepts are already encoded explicitly in the pictures. This is an

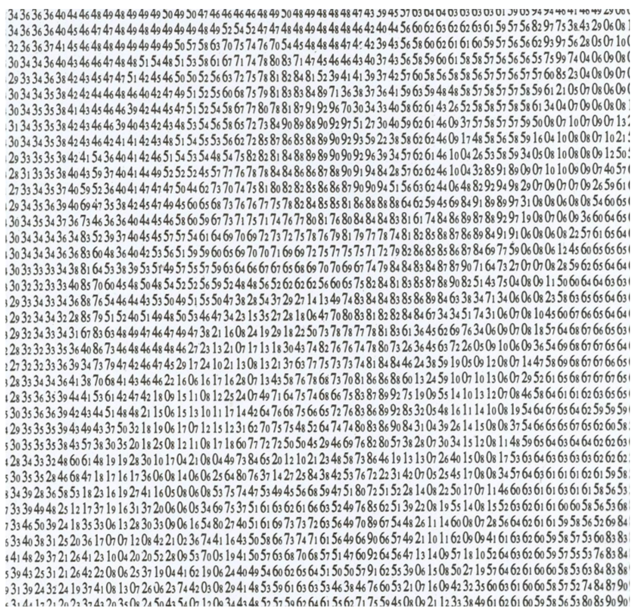


Figure 1.3: An image is merely a collection of numbers indicating the intensity values of the pixels.

illusion. A picture or an image is merely a collection of numbers. For instance, a grayscale image of 1000 $\times$ 1000 pixels is actually a 1000 $\times$ 1000 table, with each entry containing a number indicating the intensity value of the corresponding pixel. A portion of such a table is shown in Fig 1.3. For a color image, each entry contains three numbers, indicating the intensities of three basic colors, red (R), green (G), and blue (B). Nowhere in the big collection of numbers can we find explicit representations of concepts such as girl, cup, table, chair etc. Neither can we find an explicit encoding of the 3D layout of the scene and the shapes of the objects. These have to be inferred from the collection of numbers that make up the image. This is a daunting task. It is not surprising that half of the human brain is devoted to visual computation, and most of the brain's activities on cognition and reasoning are based on vision.

At this stage, we are still not entirely clear how the so-called "signal to symbol" transition is realized, i.e., how the concepts are inferred and learned from the raw images of intensities. In order to answer this question, we need to understand the correspondence between the concepts and the image patches such as the one shown in Fig. 1.3, as well as how different concepts are related to each other. We shall explore these issues in this book.

1.2 Knowledge Representation: Sets, Concepts and Models

Consider the space of all the image patches of a fixed size (e.g. 10×10) of natural scenes. These image patches reside in an image space whose dimension is the size of the image patches (i.e., the number of pixels, e.g., $10 \times 10 = 100$). We can treat each image patch as a point. Thus we have a population of points in the image space. We may consider an analogy between this population and our three-dimensional universe as illustrated by Fig 1.4 . The distribution of the mass within our universe is highly uneven. There are high concentrations of mass within the stars, while the mass spreads over the nebulas with low densities. The distribution of the population of natural image patches in the image space is also highly uneven. We need to identify, map out and catalog those high-density clusters such as stars, as well as the low-density regions such as nebulas.

Figure 1.4: A picture of the universe with mass distributed on stars (high density, low volume) and nebulas (low density, high volume). Left: the universe with galaxies, stars and nebulas. Right: a zoomed-in view.

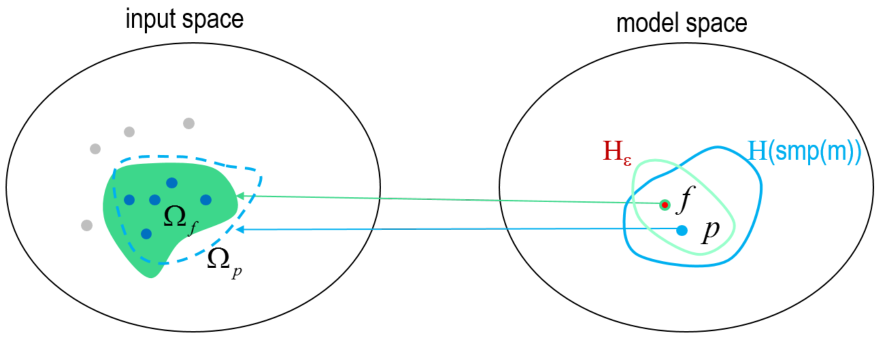


Figure 1.5: A concept corresponds to a set of image patches.

Just like a representative in US Congress represents a subset of US population, mathematically, a concept represents a subset of image patches within the whole universe of image patches. All the image patches within this set will be perceived as the same pattern, and be described by the same symbol or the same "visual word." The subset or the sub-population of the image patches that correspond to a concept can be represented mathematically by a probability distribution or a statistical model. So each concept is represented by a statistical model.

Thus we have two spaces: the space of image patches (signals) and the space of concepts (symbols), as illustrated by Fig 1.5. The left depicts the input space of image patches. The right illustrates the model space of concepts. Each concept is a point in the model space, and each concept corresponds to a set of image patches. All the image patches of a concept, despite being different in terms of their pixel intensities, correspond to the same concept and are perceived as the same pattern. For instance, in Fig 1.5, the concept f in the model space corresponds to a set Ω_f in the image space.

Suppose we observe a number of examples from Ω_f, then our goal is to learn about the concept f based on these examples. In other words, we want to recover the set Ω_f. Of course, it is impossible to recover Ω_f exactly since we do not observe all the images in Ω_f. But it is still possible to come up with an approximated estimate of Ω_f. Let us call it Ω_p, which corresponds to a point p in the model space. We hope that Ω_p is close to Ω_f, or p is close to f, for example, p is within a distance ϵ from f, measured by a pre-defined metric. In machine learning language, identifying Ω_f based on some examples from Ω_f is called learning, and these examples are called training examples.

Fig 1.6 illustrates two methods of learning the concept f by approaching Ω_f with a sequence of models.

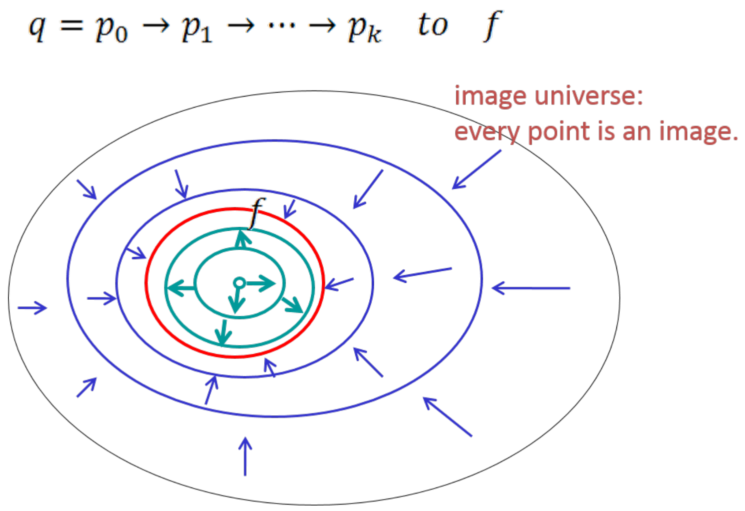


Figure 1.6: Two strategies of pursuing a concept. (1) The subset is pursued through a sequence of models by shrinking from the whole image space. (2) The subset is pursued through a sequence of models by expanding from a single point or small ball.

In the first method, we start with the whole image space, and at each step, we add a new constraint to shrink the subset. With more constraints added, we capture Ω_f from outside. In the second method, we start with a single point or small ball inside Ω_f, and at each step we expand some dimensions and fill in from inside. The reason for choosing the two pursuit strategies is quite intuitive. Some Ω_f may be of very high dimensions, thus it is efficient to capture them through constraints (by reducing the volume), whereas some other Ω_f may be of much lower dimensions, thus it is more effective to capture them by expansion (by increasing the volume).

By analogy, when a teacher is grading a final exam with the full mark being 100, for a very strong student, the teacher will start with 100, and subtract a few points here and there for errors, and then count the final score. For a very weak student, the teacher will count from 0 and add a few points here and there for credits. In practice, students at both ends are easy to grade. Students in the middle range around the pass/no-pass line need the most work.

1.3 Unification: Regimes of Representations

The 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 some stochastic textures like grasses, foil ages etc. Some concepts lie in between, such as a tiger face. Fig. 1.7 is a simple illustration of concepts with different complexities. For each concept, such as a human face, texture (e.g., foliage), or tiger, we collect a number of examples (which are aligned in the case of a human face). We then perform a principal component analysis and plot the eigenvalues in descending order. For 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 texturea, the eigenvalues stay high for an extended range, indicating that the images lie in a very high-dimensional space. For a tiger face, the pattern of the eigenvalues lies in between.

The complexity of a concept f can be measured by the log of the volume of the corresponding Ω_f. It is called entropy in statistical physics and information theory, and it is sometimes called intrinsic dimension in

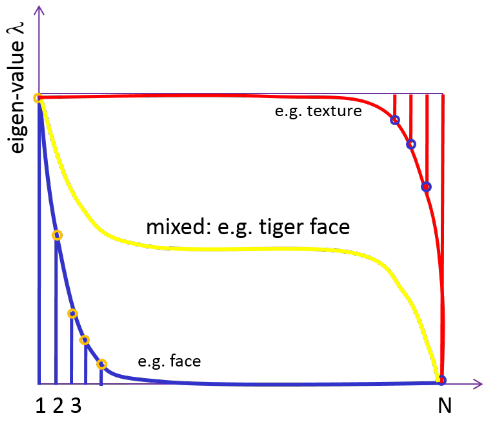


Figure 1.7: Plot of eigenvalues in decreasing order for some clusters of low dimension (blue curve) or high dimension (red curve).

mathematics, coding theory, and learning. One may also call it the degree of freedom. We can use entropy as an axis to map the concepts. The geometric patterns belong to the low-entropy regime, while the texture patterns belong to the high-entropy regime. Many object patterns lie in the mid-entropy regime. While the patterns in the low-entropy regime tend to be too simple, and the patterns in the high-entropy regime tend to be too random, the patterns in the mid-entropy regime tend to be the most informative. While the complexity of a concept f can be measured by the log-volume of the subset Ω_f as illustrated by the left panel of Fig 1.5, the informativeness of a concept can be measured by the number of parameters we need to specify the concept f in the right panel of Fig 1.5. It is an important problem to pursue a unified mathematical framework to describe all these regimes.

So far we have discussed relatively small image patches. The concepts that arise from these small image patches may be considered "atomic concepts", i.e., the simplest symbols or descriptors at the lowest layer. These 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 1.8. At the bottom layer, we have concepts for image patches, and as we shall see in later chapters, they can be described by Markov random fields for the high-entropy regime and sparse coding for the low-entropy regime. At the top layer, we have reasoning based on logic and event calculus. In the layers in between, we have recognition and cognition, and they can be described by stochastic grammars for recursive compositions. In fact, as we shall show, the stochastic grammars in the form of spatial-temporal-causal And-Or graphs can cover the whole range, and they can serve as a mathematical framework for describing the concepts at all the layers.

1.4 Conceptualization: Sets and Statistical Models

Why are the statistical models much needed in vision? What is the origin of these models? Some argued that probabilities are involved because of noise and distortions in images. This is truly a misunderstanding! With high-quality digital cameras, there is rarely noise or distortion in images anymore. Probabilities are associated with the definitions of patterns and are even derived from deterministic definitions. In fact, the statistical models are an intrinsic representation of visual knowledge and image regularities.

As we have discussed at the beginning of this chapter, the population of natural image patches can be viewed as a cloud of points in the images space, where each image patch is a point. The distribution of such a cloud of points can be described by a density function. The image patches of a certain pattern such as

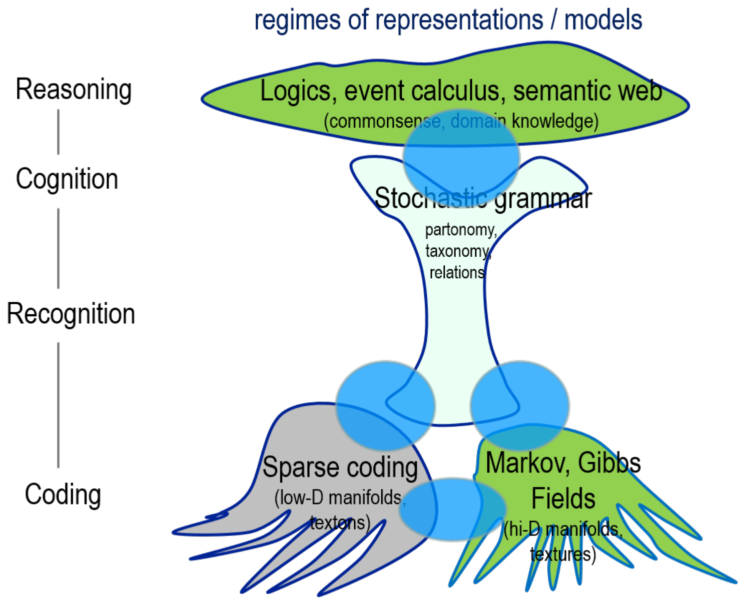


Figure 1.8: A hierarchy of concepts at increasingly abstract layers. All these concepts can be organized by And-Or graphs.

faces from a sub-population of the cloud of points. This sub-population may reside within a subset of the image space, and it can still be described by a density function.

If we randomly sample an image patch from the sub-population, then the density function of the sub-population becomes the probability density function of the random image patch. So each pattern or concept corresponds to a probability distribution on a set, which can be described by a probability density function or a statistical model.

In the literature, Grenander (1970) [66], and Fu (1973) [55] were the pioneers using statistical models for various visual patterns. In the late 1980s and early 1990s, image models become popular and indispensable when people realized that vision problems, typically the shape-from-X problems, are intrinsically ill-posed. Extra information is needed to account for regularities in real world scenes, and the statistical models represent our visual knowledge. More importantly, the statistical models are important for learning and recognizing the patterns.

2

Statistics of Natural Images

Natural images are images that are observed by humans in common daily situations. Natural images contain an overwhelming variety of structures and patterns that result from diverse physical processes. A central feature of natural images is the hierarchical compositions of objects at many different scales, much like the physical universe is hierarchically composed of subatomic particles, which compose atoms, which compose molecules, and so on. We consider captured images (video and camera) as low resolution samples from the natural image space. The space of natural images far exceeds the collection tangible photos and data created by humans through civilization. On the other hand, the natural image space is tiny in comparison to the space of all possible images.

The structure of natural images differs from the structure of non-natural images such as noise. The wide variety of visual patterns that occur in natural images share many common characteristics. For example, most natural images have simple sub-components such as flat surfaces, well-defined edges, or regular textures. The recurring structures found across natural images suggest that compact but flexible models can efficiently describe a wide array of patterns. Barlow (1961) [7] and Gibson (1966) [62] were among the earliest researchers who argued for the ecologic influence of vision perception. In modern terminology, visual perception is a machine capable of learning abstract representations from observed data. The learned representations should generalize to help understand new data. We treat previously observed images as training data, and future images as testing data.

Throughout Part 1 of this book, we discuss statistical phenomena associated with natural images with the hope of learning patterns with a common vocabulary and grammar supported by a unified mathematical and statistical foundation. In this chapter, we shall study some generic statistical properties of natural images and discuss some general issues of statistical modeling.

2.1 Image Space and Distribution

We begin with a concrete description of an image space. Consider the population of all grayscale image patches of size 10×10 pixels with each pixel intensity in the range [0, 1]. Each possible image patch is a point in the high-dimensional space $[0, 1]^{100}$. In other words, an image can be represented by a single vector with 100 elements giving the intensity of each pixel in a fixed but arbitrary order. The fixed order of the image vector does not reflect that spatial relation of the pixels but the statistical model often depends on the original pixel lattice. If the images are colored instead of grayscale, each of the three color channels has its own 10×10 patch and the image space is $[0, 1]^{300}$. The population of *natural* image patches then forms a cloud of points which is a subset of the image space. The distribution of this cloud of points follows a high-dimensional geometry which reflects the structures of natural images.

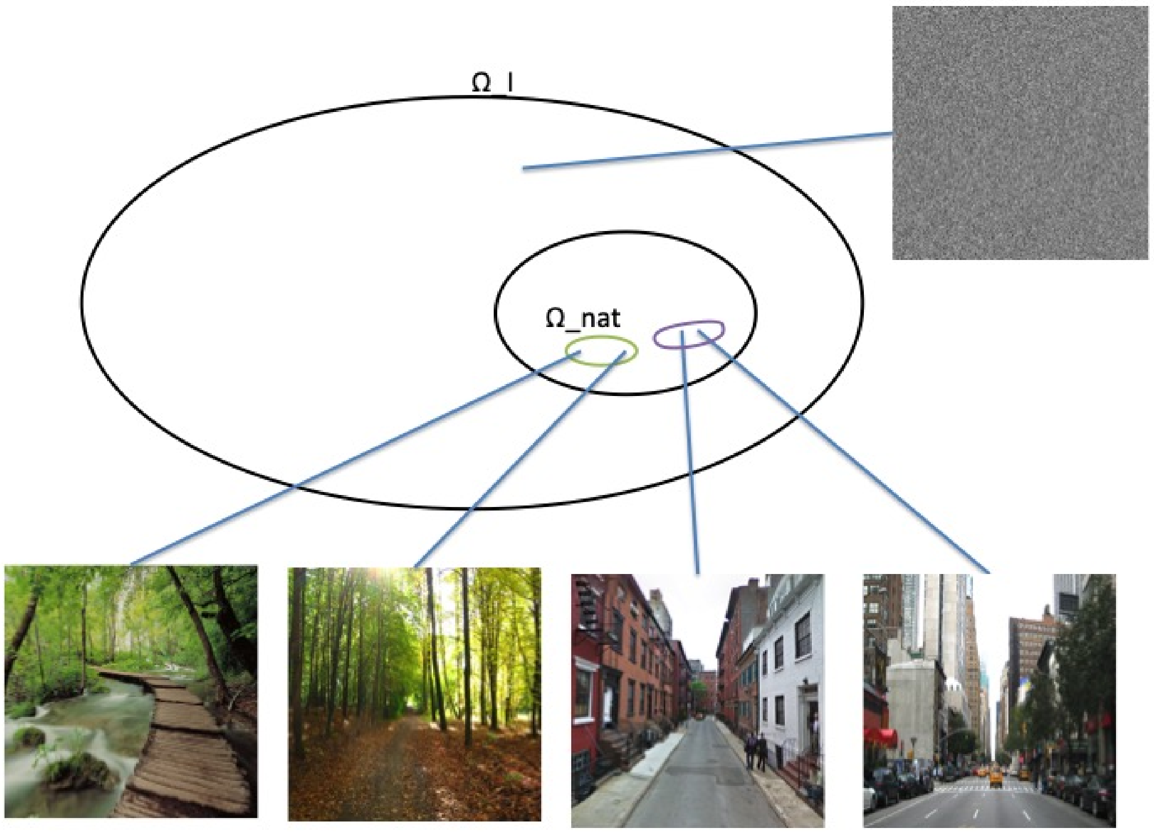


Figure 2.1: Natural image space as a subset of image space. From here we can see that $\Omega_{\text{nat}} \subset \Omega_I$. Lines denote the mapping from images to points in high dimefacensional space. Similar images lie on local manifolds of the natural image space and are separated small geodesic distances. Here we illustrate natural image one (left), two (middle left), three (mid right), and four (right) as well as an image of noise (top).

We can describe the distribution of this cloud of points by a density function, which tells us how dense (equivalently, how likely) the points are at each position (a single image) in the image space. The density of the distribution is very uneven. In some places of the image space, (e.g. a patch of a forest scene) the density will be high, while in some other places of the image space (e.g. a random sample from a high-dimensional uniform distribution), the density is virtually zero. The concentration of the density function around certain geometric regions is analagous to the distribution of mass in our universe. The density of mass is high on the stars. The density is low over nebulas. The density is practically zero over large spans of space.

Now let us introduce some notation. Let $\mathbf{I}(x)$ be an image patch defined on a square or rectangular domain, where $x \in \mathcal{D}$ indexes the location within the image. The coordinate space for x can be continuous (e.g. $\mathcal{D} = [0, 1] \times [0, 2]$ for a rectangular grayscale image with continuous axes) or discrete, as in the case where the image is composed of pixels (e.g. $\mathcal{D} = \{1, \ldots, 10\} \times \{1, \ldots, 10\}$ for a 10×10 pixel grayscale image). The coordinate space $\mathcal{D}_{\text{color}}$ of a three-channel color image is simply the Cartesian product $\mathcal{D}_{\text{color}} = \mathcal{D}_{\text{gray}} \times \{1, 2, 3\}$ of the grayscale coordinate space $\mathcal{D}_{\text{gray}}$ with the set $\{1, 2, 3\}$ that indexes the color channel. We can treat $\mathbf{I}(x)$ as a function $\mathbf{I} : \mathcal{D} \to \mathcal{V}$ that gives the intensity of each pixel in $\mathbf{I}$. For example, $\mathcal{V} = [0, 1]$ for continuous pixel intensities between 0 and 1 or $\mathcal{V} = \{0, \ldots, 255\}$ for integer-valued pixel intensities between 0 and 255.

When $\mathcal{D}$ is a finite set (i.e. for images with a finite number of pixels), we can also treat $\mathbf{I}$ as a vector if we fix an ordering for the pixels. Thus each $\mathbf{I}$ becomes a point in the image space $\Omega_I = \mathcal{V}^{|\mathcal{D}|}$ where $|\mathcal{D}|$ is the total number of pixels in the image (for colored images, each channel has its own separate pixels). For example, each 10×10 pixel grayscale image patch $\mathbf{I}$ with pixel intensities between 0 and 1 is a point $\mathbf{I} \in \Omega_I = [0,1]^{100}$. Similarly each 10×10 pixel color image with channel intensities between 0 and 1 is a point $\mathbf{I} \in \Omega_I = [0,1]^{100\times 3} = [0,1]^{300}$. A more technical definition of Ω_I is needed in the case where $\mathcal{D}$ is an infinite set, but for the majority of topics covered in the book the finite pixel-based description is sufficient.

Now consider the cloud of points $\Omega_{\text{nat}} \subset \Omega_I$ made up by *natural* image patches. The distribution of Ω_{nat} can be described by a density function $f : \Omega_I \to \mathbb{R}$ such that $f \geq 0$ and $\int_{\Omega_I} f(\mathbf{I}) d\mathbf{I} = 1$. Intuitively, for each point $\mathbf{I} \in \Omega_I$, 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 $\mathcal{N}(\mathbf{I}) \to 0$. 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 above reasoning, there is nothing probabilistic. f is just a deterministic description of a cloud of points in a high-dimensional space. Now let us bring in probability. If we randomly sample an image patch $\mathbf{I}$ from the population of natural image patches, or we randomly sample a point from the cloud of points described by the density function $f(\mathbf{I})$, then $\mathbf{I}$ is said to be a random sample or example from $f(\mathbf{I})$. Then $f(\mathbf{I})$ becomes the probability density function for the random image $\mathbf{I}$, and we write $\mathbf{I} \sim f(\mathbf{I})$. The goal of learning is to gain some knowledge about $f(\mathbf{I})$ based on a set of examples $\{\mathbf{I}_1, ..., \mathbf{I}_M\}$ that are sampled from $f(\mathbf{I})$ independently.

2.2 Information and Encoding

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

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

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

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, which is equivalent to rescaling H. For a distribution q that is uniform over a set $\Omega_q \subseteq \Omega_I$ and zero elsewhere, we have

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

where the sum over Ω_I reduces to a sum over Ω_q because density is non-zero only within Ω_q. Therefore, the entropy of a uniform distribution q is a measure of the log volume of the space Ω_q. In particular, $H(f) = \log_2 |\Omega_I|$ for the "unfocused" model that is uniform over Ω_I and $H(g) = \log_2 |\Omega_{\text{nat}}|$ for the "focused" model that is uniform over the natural image space Ω_{nat}. Later chapters will describe a more in-depth relation of entropy and learning using information theory. For now we urge the reader to form the connection between entropy and the volume of the space of natural images.

The entropy of a subset of the image space relates to the amount of information needed to represent members of the subset. Consider the space Ω_h of grayscale images that have a single horizontal black bar of fixed width against a white background. Each image can be described by a single number the 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 Ω_{hv} of grayscale images that have a single fixed-width horizontal bar and a single fixed-width vertical bar of black against a white background. All images from this set can be described by exactly two numbers representing the locations of the two bars. Suppose there are n_h positions for the horizontal bar and n_v positions for the vertical bar, so 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 of both height and width), the *entropy rate* $\bar{H}$ scales entropy relative to image size to give a per pixel entropy. For a density q distributed uniformly over Ω_q we have:

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

For example, consider the space of $n \times n$ grayscale images where $\mathcal{V} = \{0, \ldots, 255\}$, meaning each pixel value is an integer from 0 to 255. In this case $\Omega_I = \{0, \ldots, 255\}^{n^2}$ and $|\Omega_I| = 256^{n^2}$. Therefore the unfocused model $g(\mathbf{I})$ that is uniform everywhere in Ω_I has entropy rate

$$\bar{H}(g) = \frac{1}{n^2} \log_2 |\Omega_I| = \frac{n^2 \log_2 256}{n^2} = 8.$$

Thus we can encode an image random sampled from Ω_I using on average 8 bits per pixel. Since this is the maximum information needed to fully encode any given image from Ω_I, it is not possible to compress uniform random samples from the total image space.

Now if we consider an image to be a natural image $I \in \Omega_{\text{nat}}$ sampled from f then the many regularities found in natural images allows us to drastically reduce the amount of bits needed for encoding. An empirical upper bound for the average number of bits per pixel for natural images is about 0.3:

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

This reduction from the maximum value of 8 gives an intuitive measure for the amount of compression that is possible when the regular features found across previously observed natural images are used to encode new images. Compression methods work well with natural images but fail on noise images such as a QR code.

We now give a rough estimate of the size of the observed natural images space. We assume that there are 10 billion people who lived on earth at one point or another, that each of these people lived to be 100 years old, and that each human observed 20 images per second. We can estimate the volume of the natural image space as seen by humans as

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

The size of the total image universe $|\Omega_I| = 256^{n^2}$ discussed above is much larger than $|\Omega_{\text{nat}}|$ even for very small image spaces (e.g. image patches with $n = 10$ pixels height and width). Our models are meant to represent samples from the natural image space, and should reflect the regularities observed therein.

2.3 Image Statistics and Power Law

The universe is composed of ensemble structures at many different scales. From a coarse scale a structure may be perceived as an atomic entity, while from a finer scale the same structure may 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 a natural image, an object can be distinguished by regularities (such as consistent color/texture and connected surfaces) that enable perception of the object as a distinct concept. The regularities observed in natural images reflect the hierarchically structured order of the universe. Noise images on the other hand do not obey those laws. This allows one to distinguish between natural images and noise based on numerical summaries that measure structural regularities. Such summaries, also known as statistical properties, form the foundation of the more detailed understanding of images pursued in later chapters.

Our first observation about natural images is that nearby pixel intensities are typically similar in value, although occasional jumps or discontinuities can be found as well. If we consider the Earth and its various land masses, we wouldn't expect two neighboring cities to have a vast difference in their elevation (low power, high abundance). Similarly, we don't expect two neighboring pixels to have large differences in their intensity (pixel values), nor would we expect large differences in smoothness among local neighboring patches. In the case that a city is built near a mountain range, such as the great city of Kathmandu, we will indeed see a large increase in elevation between Kathmandu and the Mt. Everest base camp. This is analogous to neighboring pixels that have a large difference in intensity, which could indicate the presence of an edge or some strong feature (high power, low abundance). If we consider the entire world as our landscape, regions that encompass cities such as Kathmandu have high amplitude but are also not very abundant, as are edges in a natural image. Most landscape regions have consistent elevation and natural image landscapes follow the same trend. Finally, the elevation of two distant cities is likely to be unrelated (e.g. if we compare Kathmandu and Los Angeles's elevation), just as two distant pixels are not likely to have related intensities.

A simple measure of similarity between the intensities of two pixels of certain relative positions is covariance or correlation. For an image $\mathbf{I} \in \mathcal{V}^{|\mathcal{D}|} \subset \mathbb{R}^{|\mathcal{D}|}$ with $|\mathcal{D}|$ pixels, let us assume that we normalize the marginal mean to be 0 and 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 d which can be large when $|d| = \sqrt{d_1^2 + d_2^2}$ is small, and which will decay to 0 as $|d|$ becomes large.

An equivalent measure of relation between pixels at different locations in the image is the power spectrum of the image in the Fourier domain. The Fourier transform is a method that decomposes a signal from the spatial domain into its components in the frequency domain. In many cases (notably with discrete signals) the Fourier transform is invertible and no data is lost when the inverse Fourier Transform is used to recover the original signal. Frequency signals contain both real and imaginary components even when the original signal is real-valued, so we use the absolute value to represent the total power or energy of a signal. The power spectrum in the Fourier domain describes how relations between image pixels are distributed over different frequencies.

We can perform a Fast Fourier Transform (FFT) on an image $I \in R^{|\mathcal{D}_1| \times |\mathcal{D}_2|}$ using

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

where $|\mathcal{D}_1| = H$ is the number of pixel rows, $|\mathcal{D}_2| = W$ is the number of pixel columns, and (ξ, η) represent

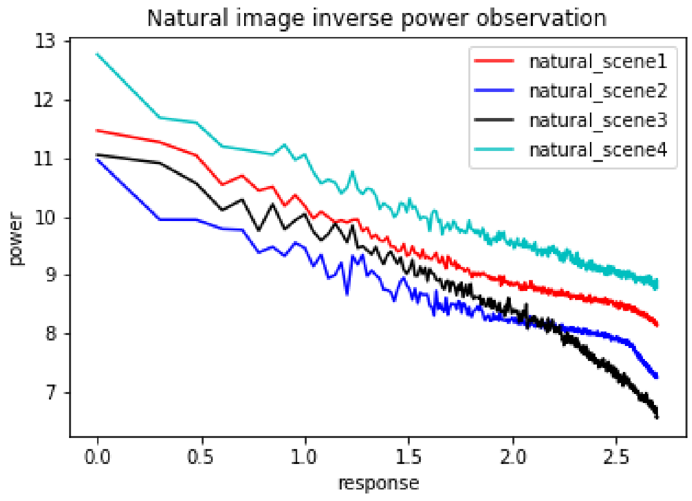


Figure 2.2: The amplitude of the Fourier coefficients vs the frequency for 4 natural images from Fig 2.1.

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 frequency (ξ, η) within the image signal $\mathbf{I}$.

An interesting empirical observation is that, for typical natural images, we approximately have

$$A(f) \propto 1/f \tag{2.2}$$

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

for all $f = \sqrt{\xi^2 + \eta^2}$. Thus the amplitude of a Fourier coefficient is inversely related to the frequency. Fig. 2.2, taken from Field (1987) [46], plots the $\log A(f)$ vs. $\log f$ for 6 natural images. The curves can be fitted by straight lines with a slope close to -1. We can use this relation to calculate the total Fourier power $A^2(f)$ for in the frequency band $[f, 2f]$ as shown in (2.4). As we increase in frequency, the total area within each frequency band increases and the amplitude decreases, thus the power remains constant because increased area is associated with decreased amplitude, providing a constant boundary for the power in each band:

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

This observation reveals that natural images contain equal Fourier power at each frequency band (octave), so natural images are scale invariant.

The power spectrum $A^2(f)$ is the Fourier transform of the covariance function $C(d)$, so both capture the same statistical property. Intuitively $A^2(f)$ tells us the fluctuations of the image content at different scales or frequencies. The spectrum and the covariance are said to capture the second order moments of the image distribution. Such second order properties 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) [135] that a Gaussian Markov Random Field (GMRF) model below has exactly

$1/f$-Fourier amplitude (see Section 3.2.3):

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

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

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

are discrete approximations of the gradients.

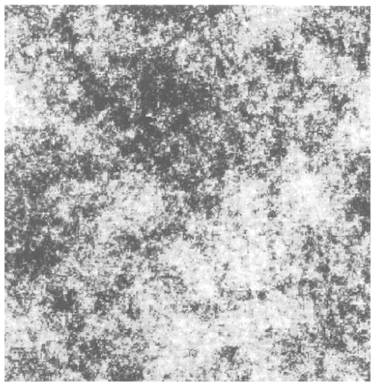


Figure 2.3: A random sample from the Gaussian Markov random field model.

Since $p_{1/f}(\mathbf{I};\beta)$ is a Gaussian model, one can easily draw a random sample $\mathbf{I} \sim p_{1/\mathrm{f}}(\mathbf{I};\beta)$. Fig. 2.3 shows a typical sample image by Mumford (1995). This model matches the local regularity of natural image statistics and only that. It can be considered a natural image model that has only one constraint, making it slightly better than noise. Large values of β lead to samples with large patches of similar intensity and small values of β will have more variation in nearby pixels. The limiting case $\beta \to \infty$ concentrates all mass on the zero iamge $\mathbf{I} = 0$. From this, we can observe that certain statistics found in natural image also are found for this model, but these statistics (covariance and power spectrum) do not contain enough complexity to capture key features needed for natural image modeling and inference.

The power spectrum can also be pooled by the Gabor filters centered at different frequencies. The Gabor filters are sine and cosine waves multiplied by elongate Gaussian functions. The filter response is a localized Fourier transform. Let F be a Gabor filter whose sine and cosine waves have a frequency ω. Let $[F * \mathbf{I}](x)$ be the Gabor filter response at location x. Then $[F * \mathbf{I}](x)$ measures the frequency content of $\mathbf{I}$ around frequency ω at location x. Due to the spatial localization of F, F also responds to sine and cosine waves whose frequencies are close to ω. Thus if we pool $\sum_x |F * \mathbf{I}|^2/|\mathcal{D}|$, it will be the average of the power spectrum within the band of frequencies that F responds to, such as the one in equation (2.5). The inverse power law says that $\sum_x |F * \mathbf{I}|^2/|\mathcal{D}|$ remains scale invariant, i.e., it is constant for different ω. Furthermore, this is a second order property that can be reproduced by the Gaussian distribution.

2.4 Kurtosis and Sparsity

As mentioned above, the second order properties such as the covariance function, the power spectrum, or the average of the squared filter responses, can be fully reproduced by a multivariate Gaussian distribution

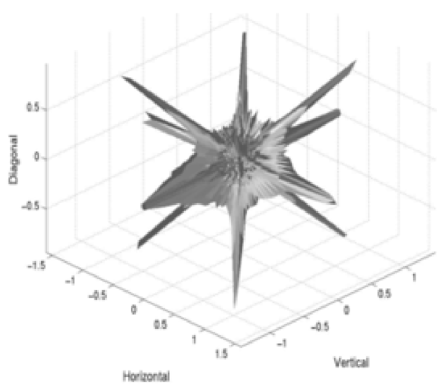


Figure 2.4: The distribution of natural images may have many low dimensional spikes.

with a matching variance-covariance matrix. We can think of the Gaussian distribution as a cloud of points. For Gaussian distributions, this cloud of points form an ellipsoid shape.

The ellipsoid shaped Gaussian distribution is rather dull. It cannot be expected to capture the presumably highly sophisticated shape of the natural image distribution. For instance, the distribution of natural images may have many low dimensional spikes, as illustrated by Fig. 2.4. Hence, researchers have been looking for patterns of deviations from the Gaussian distribution, i.e., searching for the non-Gaussian features.

For example, the study on natural image statistics extends from covariances to histograms of filter responses (e.g. Gabor filters). While the covariances only measure second order moments, the histograms include higher order information, such as skewness and kurtosis.

More formally, let X denote a random variable with mean μ and variance σ^2, we can normalize X by the transform $(X-\mu)/\sigma$. Skewness is the third order moment $\mathrm{E}[((X-\mu)/\sigma)^3]$. It measures the asymmetry of the distribution. Kurtosis is the fourth order moment $\mathrm{E}[((X-\mu)/\sigma)^4]$. A histogram with heavy tails has high kurtosis. When we measure skewness and kurtosis, the first order moment μ and the second order moment σ^2 only serve to normalize the random variable (zero mean and unit variance).

We utilize the idea of a 1D marginal projection in order to gain understanding of a higher dimensional space such as Ω_{nat}. A histogram can be considered a marginal projection because it contains statistics in one dimension (line). The study of the histograms leads to an interesting observation. The histograms of Gabor filter responses on natural images, like that shown in eq 2.6, where F is a Gabor filter and $f_n(I)$ is a "focused model" (zero outside of the space of natual images) have high kurtosis (Field, 1994), and are heavy tailed. That is, they have tails that are heavier than the Gaussian distributions. This reveals that natural images have high-order non-Gaussian structures. Gabor filters are primarily used to detect orientation but other filters are also acceptable such as gradient filters, Haar filters, etc.

The histograms from eq 2.6 can also be visualized as shown in Fig 2.5. Here we observe an image I located in $\Omega_{\text{nat}} \subset \Omega_I$, a high dimensional space. F is a local filter, one that has a response with a portion or patch of image I represented by each point. The inner product $< F, I >$ will be zero when $F \perp I$. The two filter responses illustrate the binning operation performed in eq 2.6. We can observe high kurtosis due to the vast amount of orthogonality between F and I. Any filter (F) response will show the same high kurtosis heavy tail structure.

$$h(a) = \iint_{\Omega_I} f(I)\delta(< F, I >\in a)dI \tag{2.6}$$

The high kurtosis in the image statistics is only a marginal evidence for hidden structures in natural

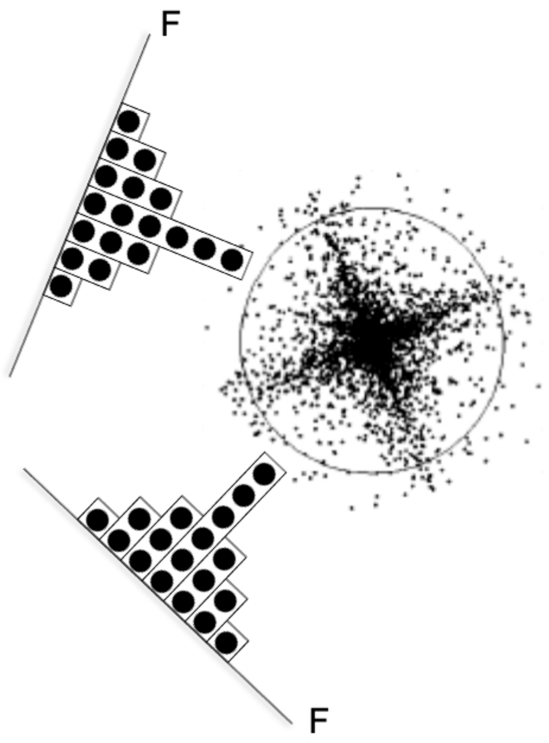


Figure 2.5: Histogram as 1D marginal projection

scenes. A direct way for discovering structures and reducing image redundancy is to transform an image into a superposition of image components. For example, Fourier transform, wavelet transforms [125], and various image pyramids [172] for generic images, and principal component analysis for some specific ensembles of images, such as face images.

The transforms from image pixels to other linear basis such as Fourier, wavelets, or PCA basis, achieve two desirable properties. The first is variable decoupling - coefficients of these bases are less correlated or become independent in ideal cases. The second is dimension reduction - the number of basis vectors for approximately reconstructing an image is often 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 a mathematical tool for decomposing images is harmonic analysis. Harmonic analysis decomposes various classes of functions (ie mathematical spaces), or signals as a superposition of basic waves/ basis systems.

Obviously the population of natural images is quite different from those functional classes. Therefore, the image components must be adapted to natural images. This leads to an inspiring idea — sparse coding with over-complete basis or dictionary by Olshausen and Field, 1996 [144]. With over-complete basis, an image may be reconstructed by a small (sparse) number of basis vectors in the dictionary. This often leads to 10-100 folds of dimension reduction. For example, an image of 200×200 pixels can be reconstructed approximately by about $100 - 500$ base images. Olshausen and Field then learned the over-complete dictionary from natural images. This sparse coding idea reflects the assumption that the point cloud formed by the population of natural images has many low dimensional spikes as illustrated in Fig. 2.4.

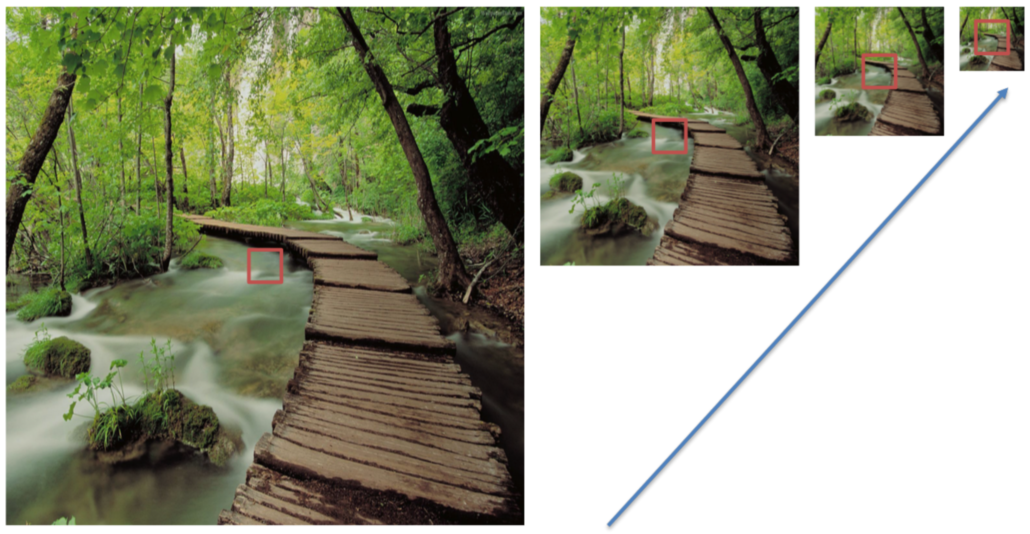


Figure 2.6: Natural image 1 scaled through 1x, 2x, 4x, 8x downsampling. Athough each filter kernel (shown in red) captures more information for each downsampled scale, the statistics remain invariant.

2.5 Scale Invariance

In addition to the heavy tailed histograms of Gabor filter responses, another interesting observation reported by Ruderman (94) [162] and Zhu and Mumford (97) [223] is that the histograms of gradient filtered images are consistent over a range of scales, see Fig. 9.4.

Specifically, for a natural image $\mathbf{I}$, we build a pyramid with a number of n scales, $\mathbf{I} = \mathbf{I}^{(0)}, \mathbf{I}^{(1)}, ..., \mathbf{I}^{(n)}$. $\mathbf{I}^{(s+1)}$ is obtained by averaging every block of 2×2 pixels in $\mathbf{I}^{(s)}$. The histograms of gradients $\nabla_1 \mathbf{I}^{(s)}(x)$ for natural images are plotted in Fig. 9.4 (top panel) for three scales $s = 1, 2, 3$ (shown in Fig 2.6). Fig. 9.4 (bottom panel) for gaussian noise image over the same scales.

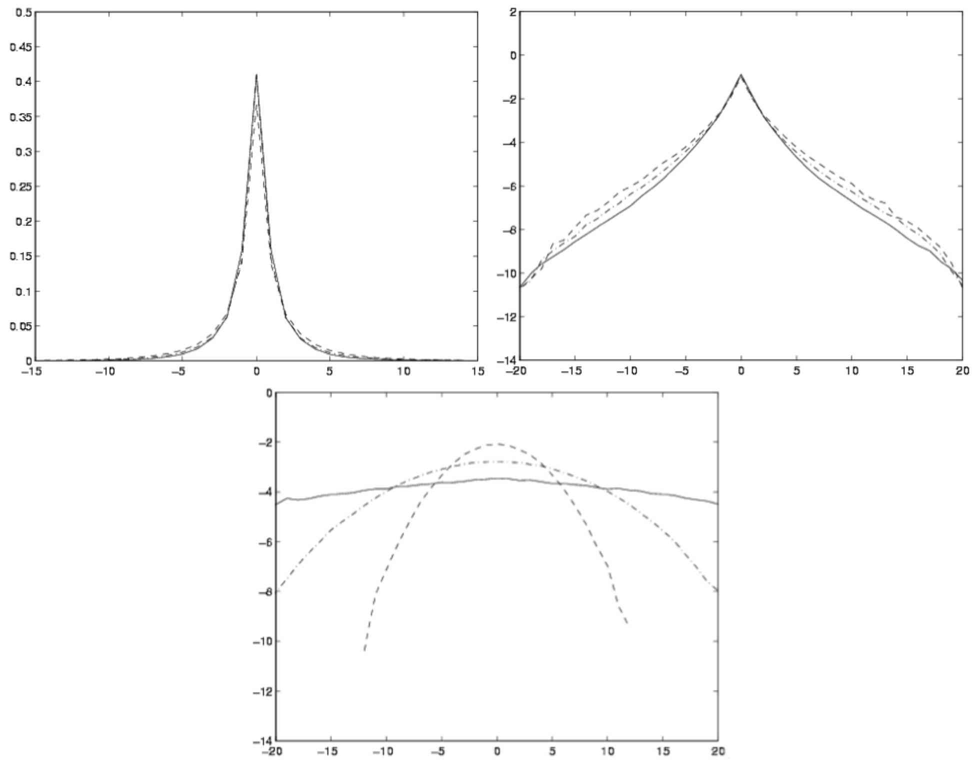


Figure 2.7: Top: histograms of gradients of natural image 1 from Fig 2.1 at different scales Right: log plot. Bottom: log plot of histograms of gradients of Gaussian noise at different scales. From this we can observe that natural image statistics such as high kurtosis and heavy tail occurs in natural image, and does not occur in noise, and that these statistics persist over different scales.

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

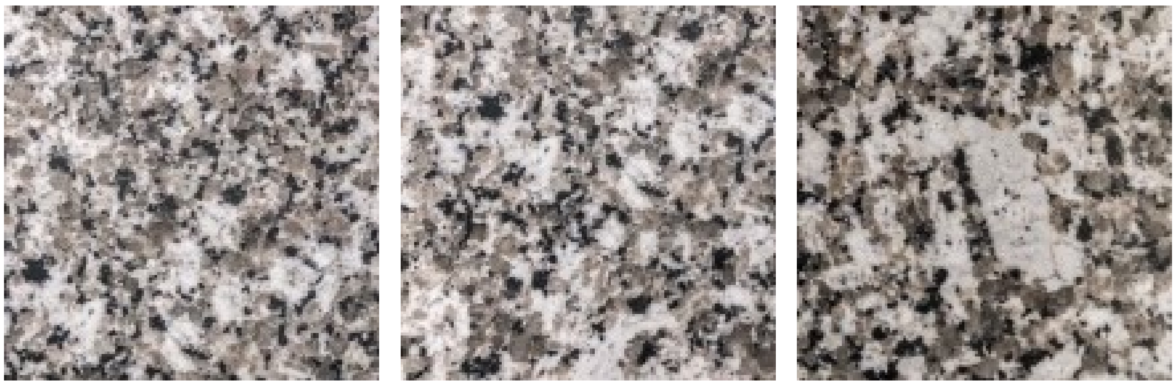

(a) (b) (c)

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

Imagine a scenario when you walk into a store and ask to purchase some marble tiles for your new home. You try to search for the most consistent pattern on your tile so your new floor looks uniform and each piece looks indistinguishable from one another. Suddenly, you notice a piece of marble, as shown in Figure 3.1a and 3.1b, whose patterns you are quite satisfied with. However, as you continue scanning this marble piece, you suddenly catch attention of an area, as in Figure 3.1c. You continue to ask yourself one question, "Is this texture patch consistent with the rest of the marble?" "What features should a texture have to be judged different from another texture patch by a human?" These questions may seem ostensibly simple to answer, but it is fundamentally difficult to exactly define distinguishing features that set a texture apart from another.

Differentiating differences in texture patterns seems an intuitive and easy task for humans to complete, 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 tell apart the defective patterns? How are texture patterns represented in human brains in general? This is a question that troubled psychologists for many years until Julesz conducted formal research on texture modeling.

In his seminal paper of 1962 [95], Julesz initiated research on texture by asking 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. 4.2, the texture pattern formed by the $+$ signs pops out from the surrounding pattern at the first glance (the 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.

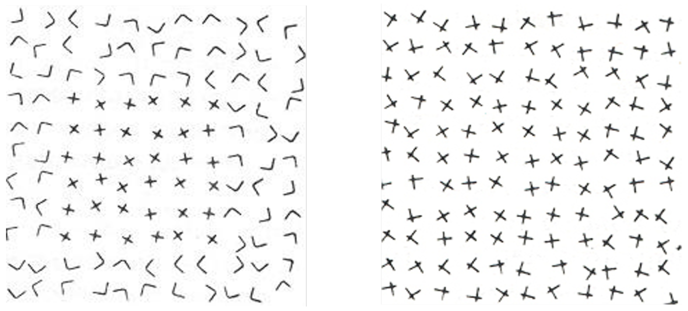


Figure 3.2: Julesz texture pairs: the texture pattern formed by the + signs pops out from the surrounding pattern in the left panel, but it merges with the surrounding pattern in the right panel.

The first general texture model was proposed by Julesz in the 1960's. 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 [95]. A key drawback of this model is that, on the 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 [35].

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

$$\Omega_{\mathbf{I}} = \{\mathbf{I} : \mathbf{H_i}(\mathbf{I}) = \mathbf{h_i}, i = 1...K\} \tag{3.1}$$

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 2-gon statistics conjecture. Examples include co-occurrence matrices, run-length statistics, sizes and orientations of various textons, cliques in Markov random fields, as well as dozens of other measures. All these features have rather limited expressive power.

Remark: Here, it is worth noting that the modeling of texture concerns not merely with the data and distributions in the image space, but depends on human perception which is influenced by task and purpose.

In Julesz's quest, the criterion for judging whether two texture images are from the same category depends on human vision systems, which are trained for various tasks. Julesz carefully reduces the task-dependence by testing in a pre-attentive stage with no specific purpose, neverthless the human vision system has been trained ahead of the experiments. Later experiments show that the human vision can learn and adapt to tell apart texture images after long training. This task-dependence sets apart models in vision from models in physics.

In the coming chapters, let's walk through some major quest for the features and statistics describing texture patterns.

3.2 Clique-Based Gibbs Models & MRF

One area of pursuing texture features is statistical modeling, which characterizes texture images as arising from probability distributions on random fields [27]. These modeling approaches involve only a small number of parameters, thus provide concise representation for textures. More importantly, researchers pose texture analysis as a well-defined statistical inference problem. The statistical theories enable us not only to make inference about the parameters of the underlying probability models based on observed texture images, but also to synthesize texture images by sampling from these probability models. Therefore, it provides a rigorous way to test the model by checking whether the synthesized images have similar visual appearances to the textures being modeled.

But usually these models are of very limited forms, hence suffer from the lack of expressive power.

3.2.1 Markov Random Fields (MRF)

Markov Random Fields (MRF) models were popularized by Besag 1973 [13] for modeling spatial interactions on lattice systems and were used by Cross and Jain 1983 [27] for texture modeling. An important characteristic of MRF modeling is that the global patterns are formed via stochastic propagation of local interactions, which is particularly appropriate for modeling textures since they are characterized by global but not predictable repetitions of similar local structures.

In MRF models, a texture is considered 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 a neighborhood system of $\mathcal{D}$, which is a collection of subsets 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 called 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.

Concretely, MRF models are undirected graphs of random variables following Markovian properties. The right figure in Figure 3.3 shows a simple general MRF, where, for example, node B's neighbors contain C, D and E, and nodes B, D, and E form a three-node clique.

Formally, the following gives a definition of MRF distributions.

Definition. *$p(\mathbf{I})$ is an MRF distribution with respect to $\mathcal{N}$ if $p(\mathbf{I}_s \mid \mathbf{I}_{-s}) = p(\mathbf{I}_s \mid \mathbf{I}(\mathcal{N}_s))$, where $\mathbf{I}_{-s}$ 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, MRF distribution is closely related to Gibbs distribution, defined below.

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

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

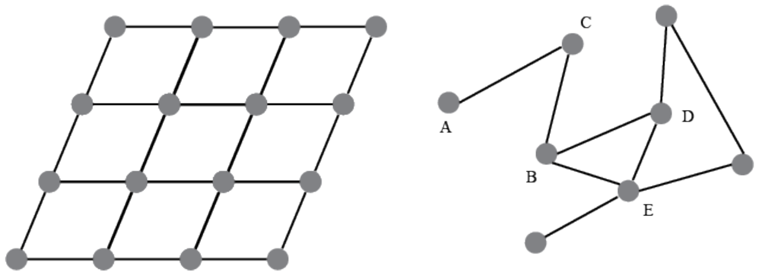


Figure 3.3: On the left is an example lattice structure of a MRF. On the right is a toy example of a general MRF where A, C are neighbors of each other, thus forming a two-node clique and B, D, E form a three-node clique.

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 λ_C for them to be uniquely determined.

In fact, MRF distribution is equivalent to Gibbs distribution. The Hammersley-Clifford theorem establishes their equivalence [13]:

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

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

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

Often enough, a texture is considered as a MRF in a lattice system with each pixel represented by a node, as shown on the left of Figure 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

To talk about MRF models with pair potential, we look at two important models developed in history, Ising-model and Potts-model.

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

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

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

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

where the neighborhood is usually of order less than or equal to three pixels, and some further restrictions are usually imposed on g for $p(\mathbf{I}_s \mid \mathbf{I}_{-s})$ to be a linear regression or the generalized linear model.

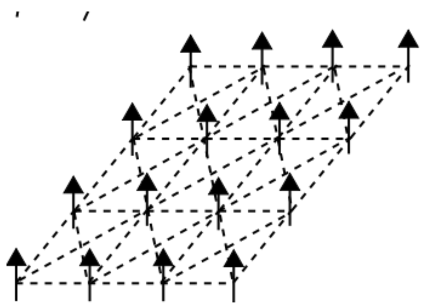


Figure 3.4: Graphical view of Ising Model with all positive spin.

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

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

$$H(\mathbf{I}) = -\sum_{<s,t>} \beta \mathbf{I}_s \mathbf{I}_t - \sum_t \alpha \mathbf{I}_t \tag{3.4}$$

where β represents the strength of magnetization and dictates electron interactions and α represents strength of external field on each electron.

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

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

where 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 is. An example sampling of Ising Model using different β is shown in Figure 3.5.

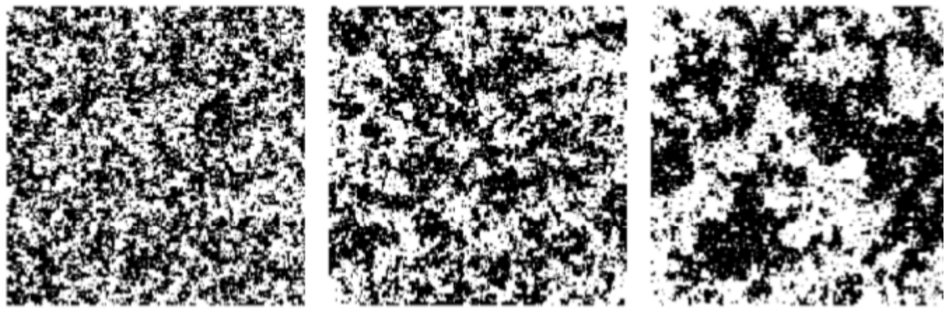


Figure 3.5: Example sampling of Ising Model with different mean and variance. From left to right, β is 0.35, 0.40, and 0.43, respectively.

Furthermore, Potts Model is a generalization to Ising Model in a lattice system. Instead of having binary state, each particle has n spin angles, i.e. $\mathbf{I}_s \in \{\theta_0, \theta_1, ..., \theta_n\}$. Therefore, the energy Hamiltonian is given by

$$H(\mathbf{I}) = -\sum_{<s,t>} \beta\delta(\mathbf{I}_s, \mathbf{I}_t) - \sum_t \alpha \mathbf{I}_t \tag{3.6}$$

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

The advantage of pair-potential models such as Ising and Potts Model is that the parameters in the models can be easily inferred by auto-regression, but they are severely limited in the following two aspects: i) the cliques are too small to capture features of texture, ii) the statistics on the cliques specifies only the first-order and second-order moments (e.g. means and covariances).

However, many textures have local structures much larger than three or four pixels, and the covariance information or equivalently spectrum cannot adequately characterize textures, as suggested 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 [35]. Moreover, many textures are strongly non-Gaussian, regardless of neighborhood size.

The underlying cause of 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 the local interactions.

3.2.3 Gaussian Markov Random Fields (GMRF)

An early statistical model for texture patterns is 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 synthesize 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 $x \sim \mathrm{N}(\mu, Q^{-1})$ 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 dependence satisfies $x_k \perp x_\ell \,|\, x_{-k\ell} \Leftrightarrow Q_{k,\ell} = 0$ as can be seen from direct inspection of the multivariate Gaussian density. Therefore $x_k \,|\, x_{-k} \sim x_k \,|\, \mathcal{N}(x_k)$ where $\mathcal{N}(x_k) = \{x_\ell : Q_{k,\ell} \neq 0, k \neq \ell\}$ is the neighborhood of x_k. 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. The edges often represent nodes that are spatially related, such as nearby pixels within an image.

Consider an $N \times M$ image $\mathbf{I} \sim \mathrm{N}(0, \beta^{-1}Q^{-1})$ where $\mathbf{I}(x, y)$ denotes pixel at location (x, y) (Note: this is different from the previous graph structure notation where $\mathbf{I}_s$ denotes node s in $\mathbf{I}$.) and $\beta > 0$ 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

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

and the conditional expectation for each pixel

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

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

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

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 density (3.8). Recall that the Fourier transform $\hat{I}(\xi,\eta)$ of an image $\mathbf{I}(x,y)$ is given by

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

Algebraic manipulation shows that $F(\nabla_x \mathbf{I}) = 2\pi i \xi \hat{I}$ and $F(\nabla_y \mathbf{I}) = 2\pi i \eta \hat{I}$. 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 L_2 functional norm is preserved by the Fourier transform. Bringing all this together shows

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

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

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

independently of other states of $\hat{I}$. This shows that $\hat{I}(\xi,\eta)$ is a Gaussian with parameters

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

Therefore $E[|\hat{I}(f)|^2]^{1/2} \propto \frac{1}{f}$ 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 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 dynamic

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

is equivalent to heat-diffusion equation

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

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

Using Euler-Lagrangian equation for two variable,

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

where $J(f) = \int_y \int_x L(x,y,f,f_x,f_y)dxdy$ 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

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

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

Despite the interesting connections to basic properties of natural images and 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 (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 (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.

3.3 Filters for Early Vision

Due to 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 retina, to V1, to 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 visual system decomposes the retinal image into a set of sub-bands, which are computed by convolving the image with a bank of linear filters followed by some non-linear procedures [12]. Considering again the definition $\Omega(\mathbf{h}) = \{\mathbf{I} : \mathbf{H_i}(\mathbf{I}) = \mathbf{h_i}, i = 1...K\}$, it is now natural to use the marginal statistics (histograms) after convolving linear filters with an image to represent feature $\mathbf{H_i}$ of an image.

Filtering refers to a process of changing pixel values of a given image and its goal is to modify images by extracting valuable information such as clusters and edges. Figure 3.6 provides an example of typical image filtering.

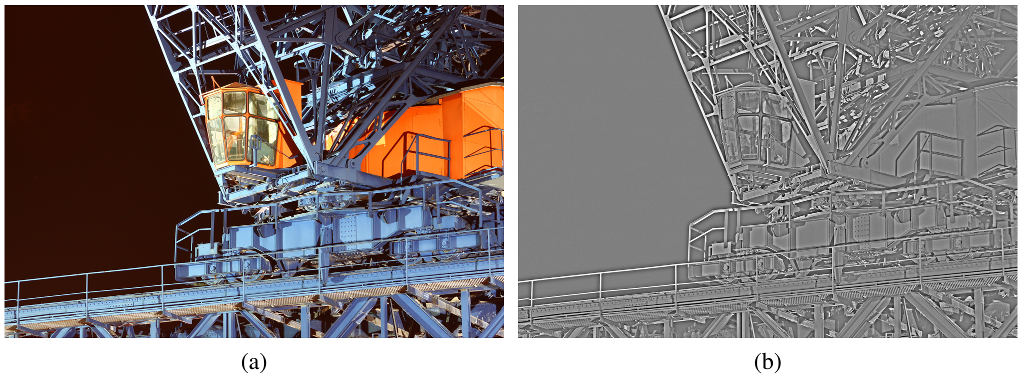


Figure 3.6: (a) presents an image and (b) presents a filtered version of the same image.

As 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, however, it is important to understand how filters are applied to images.

One method of linear filtering is correlation, which involves a weighted combination of all pixels in a small neighborhood using filters. 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 to represent pixel intensity. Suppose also that filter F is defined on an $M \times M$ lattice much smaller than that of $\mathbf{I}$. Correlation is computed as

$$\mathbf{I}(x, y) \otimes F = \sum_{(k,l)} \mathbf{I}(x + k, y + l)F(k, l) \tag{3.9}$$

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

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

$$\mathbf{I}(x, y) * F = \sum_{(k,l)} \mathbf{I}(x - k, y - l)F(k, l) \tag{3.10}$$

where $*$ denotes the convolution operation. It appears that convolution is correlation with both axes flipped. The motivation to use convolution instead of correlation is that it is in associative in addition to being

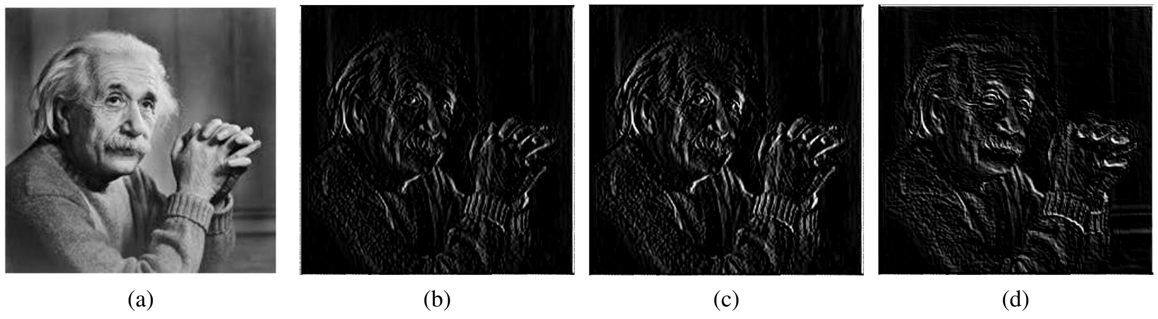


Figure 3.7: An image filtered by a Prewitt, Sobel and Roberts filter, respectively.

commutative and distributive. That is, 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. Detailed proof on convolution's mathematical properties is not of interests of this book, but to see why convolution is associative, notice that Fourier transform of convolution is a product in the frequency domain, which is clearly associative.

Knowing the basic operations for image processing, let's 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

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

To only account for horizontal edges or vertical edges, we simply use $[\frac{\partial \mathbf{I}}{\partial x}, 0]$ or $[0, \frac{\partial \mathbf{I}}{\partial y}]$. The gradient direction θ, as in calculus, is

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

and gradient strength is

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

In practice, however, images consist of discrete pixels, so we need a discrete approximation of image derivatives. Note that the definition of derivatives w.r.t x-coordinate is

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

and its discrete counterpart

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

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

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

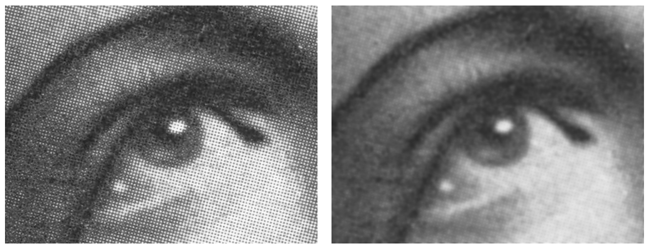

(a) (b)

Figure 3.8: An image filtered by a Gaussian filter.

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

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

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

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

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

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

Both Sobel and Prewitt filter above detect edges in the x-direction and detection in the y direction is similar with the filter weights transposed. However, Sobel filter are more commonly used because it provides a smoothing in the direction of perpendicular to the edge detection direction. Details on smoothing will be provided in the next section. Images filtered by Prewitt may suffer from noise in the original image. Different from the other two, Roberts filter detects edge in the diagonal direction instead. Figure 3.7 shows an image filtered by the filters mentioned above.

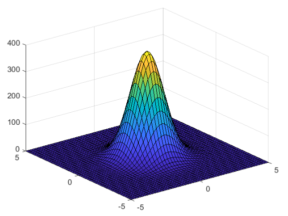
3.3.3 Gaussian Filter

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

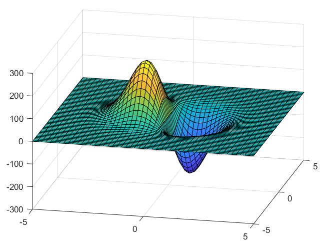
$$G(\vec{x} \mid \Sigma) = \frac{1}{\sqrt{(2\pi)^n |\Sigma|}} \exp\left\{ -\frac{1}{2}(\vec{x} - \vec{x_0})^\intercal \Sigma^{-1} (\vec{x} - \vec{x_0}) \right\}$$

where G denotes Gaussian function and $\vec{x}$ is an n-dimensional variable centered at $\vec{x_0}$ and Σ 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

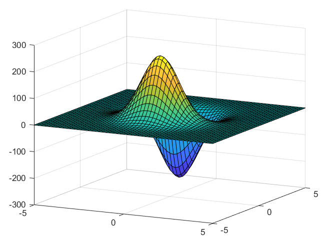
$$G(x, y \mid x_0, y_0, \sigma_x, \sigma_y) = \frac{1}{2\pi\sigma_x\sigma_y} e^{-((x-x_0)^2/2\sigma_x^2 + (y-y_0)^2/2\sigma_y^2)}$$



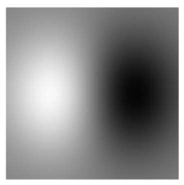
(a)



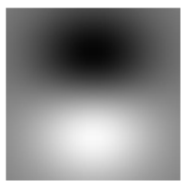
(b)



(c)



(d)



(e)

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.

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 σ allows one to vary width of the Gaussian distribution, controlling how much smoothing one wants. This corresponds to a larger filter size with each convolution accounting for more neighboring pixels. Note that Gaussian filtering is a form of weighted sum with the highest weight at the center, this smoothes out noises in close vicinity. A Gaussian smoothed image is shown in Figure 3.8.

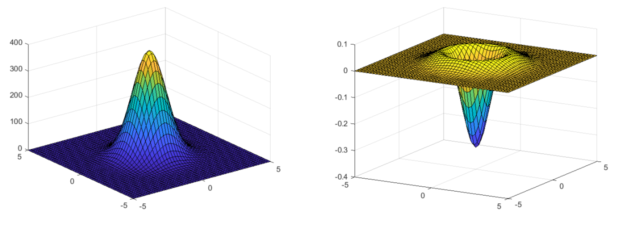
3.3.4 Derivative of Gaussian & Laplacian of Gaussian Filter

Coming back to edge detection, a major problem Sobel and Prewitt filters suffer from is that they are still very sensitive to noises due to hard subtraction of neighboring values. It is therefore common to first conduct Gaussian smoothing before applying image gradients. Since convolution is associative, it is true that we can directly apply gradient operation to Gaussian before applying to the image itself.

This suggests that we need to calculate first derivative of Gaussian, and Figure 3.9 shows first derivatives w.r.t both x and y-coordinate and their corresponding filters.

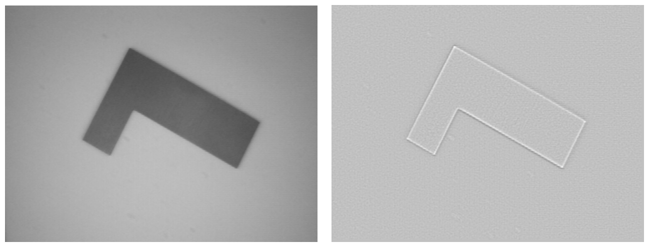
Convolving the above filters with an image allows for both smoothing and edge detection. It is less sensitive to noise than Prewitt or Sobel.

For edge detection purposes, first derivatives of Gaussian are commonly used, but they are still to some degree not desirable due to thickness of the detected edges induced by σ in Gaussian function. Second derivative, on the other hand, shows the edges with zero intensity since 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 second derivative, Laplacian $\Delta = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$ is most commonly used.



(a) (b)

Figure 3.10: Density plots of Gaussian and second derivative of Gaussian probability density function.



(a) (b)

Figure 3.11: Example filtering by Laplacian of Gaussian.

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

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

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

$$LoG(x, y \mid x_0, y_0, \sigma_x, \sigma_y) =$$

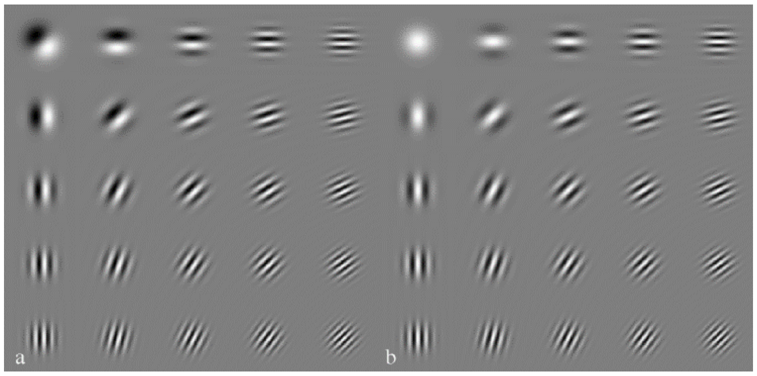


Figure 3.12: Gabor wavelets in different orientations and frequencies.

$$-\left[\frac{1}{2\pi\sigma_x^3\sigma_y}\left[1-\frac{(x-x_0)^2}{\sigma_x^2}\right]+\frac{1}{2\pi\sigma_x\sigma_y^3}\left[1-\frac{(y-y_0)^2}{\sigma_y^2}\right]\right]\exp\{-(\frac{(x-x_0)^2}{2\sigma_x^2}+\frac{(y-y_0)^2}{2\sigma_y^2})\}$$

Figure 3.10 shows probability density function of Gaussian distribution and Laplacian of Gaussian distribution. Note that the uphill surrounding a "valley" indicates that after convolution with an image an edge whose low-intensity side will rise to have high intensity before dropping to zero and down to low intensity before rising again to higher intensity on the other side of the edge. An example filtering using Laplacian of Gaussian is shown in Figure 3.11.

3.3.5 Gabor Filter

Many efforts have been spent on modeling radially symmetric center-surround retinal ganglion cell. One such simple models is the Laplacian of Gaussian introduced above with $\sigma_x = \sigma_y = \sigma$.

As retinal ganglion cells respond to specific frequency bands, and Gabor filters provide a feasible way to extract from an image texture patterns under specific frequencies [30]. In addition to being frequency-selective, Gabor filters are also directional-selective. Mathematically, Gabor filter is a pair of cosine and sine waves with angular frequency ω and η and amplitude modulated by the Gaussian function. Its general form is

$$F_\omega(x,y) = G(x,y \mid x_0,y_0,\sigma_x,\sigma_y)\cdot S(x,y \mid \omega,\eta) \tag{3.11}$$

where $G(x,y \mid x_0,y_0,\sigma_x,\sigma_y)$ 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_ω defines a Gabor filter that matches image regions with frequency ω in the x-direction and frequency η in the y-direction. High frequency filters will easily match high frequency patterns whereas low frequency filters match low frequency patterns.

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

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 filter $|(F * \mathbf{I})(x,y)|^2$. 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. For example, i) given a bank of filters, how to choose the best set of filters? Especially when some of the filters are linear while others are non-linear, or the filters are highly correlated to each other,

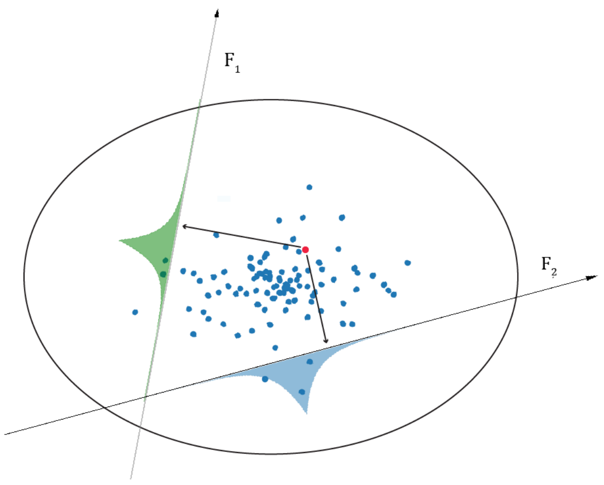


Figure 3.13: Suppose every point is an image and the red dot represent the image in question. Two filters are seen as axes piercing through the image space. The features described by these two filters are represented by this image's projection (convolution) against the two axes.

ii) after selecting the filters, how to fuse the features captured by them 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 inspirations 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 proposes 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 [226]. Before continuing, however, you are encouraged to go over Chapter 8 and understand Maximum Entropy, Minimum Entropy, and Minimax Entropy Principles as we will be using them in this chapter for derivation.

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 $\phi \in S = \{\phi^{(\alpha)}, \alpha = 1, 2, ..., K\}$ and K is the number of features in question. In FRAME, each feature is typically a vector of (either normalized or unnormalized) histograms of activations after image $\mathbf{I}$ is convolved with a selected filter. As a reminder, a filter can be seen as a low-dimensional space piercing through the image space and convolution of an image with the filter can be seen as the projection of the image onto this low-dimensional space. The projected histograms are the marginal distributions of the image, and we aim to match "enough" marginal distributions to model the actual image. An intuitive drawing is presented as Figure 3.13.

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

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

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

$$\Omega_{\mathbf{I}} = \{\mathbf{I} : \mathrm{E}_p[\phi^{(\alpha)}(\mathbf{I})] = \mu_{obs}^{(\alpha)}, \;\; i = 1...K\}$$

given a model distribution $p(\mathbf{I})$. Similarly, we can define the set of distributions that match the statistics as

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

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

This pursuit process is two-fold, one being selecting K suitable filters from a given set of filters to completely describe the most distinctive features of a texture, and the other being finding the best parametrized distribution to explain the current statistics given a fixed set of filters. This parametrized distribution can be written as $p(\mathbf{I}; \Lambda; S)$. Filter selection and distribution matching is an iterative process, reminiscent of coordinate descent, in minimizing differences between the true distribution and model distribution. Note that as we try to model pre-attentive processes in human brains when detecting textures, we generally neglect training of filters and use pre-trained filters.

We make use of the Maximum Entropy Principle to pursue the best parameters Λ when the set of features (parametrized by a set of filters) to be matched is fixed. The parameterized distribution is constrained to match histograms of the filter activations. 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.

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

$$p(\mathbf{I}; \Lambda | S) = \frac{1}{Z(\Lambda)} \exp\{-\sum_{\alpha=1}^{K}\langle\lambda^{(\alpha)}, \phi^{(\alpha)}(\mathbf{I})\rangle\}, \tag{3.12}$$

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

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 simplications derived in 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:

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

The matched statistics gives an equivalence class of images $\Omega(\mathbf{h}) = \{\mathbf{I} : \mathbf{H_i}(\mathbf{I}) = \mathbf{h_i}, i = 1...K\}$ in which each $\mathbf{I} \in \Omega_{\mathbf{I}}(\mathbf{h})$ is similar in the sense that their projected histogram statistics are similar. Therefore, the more filters are chosen, the more constrained the distributions are, and fewer images there are in the

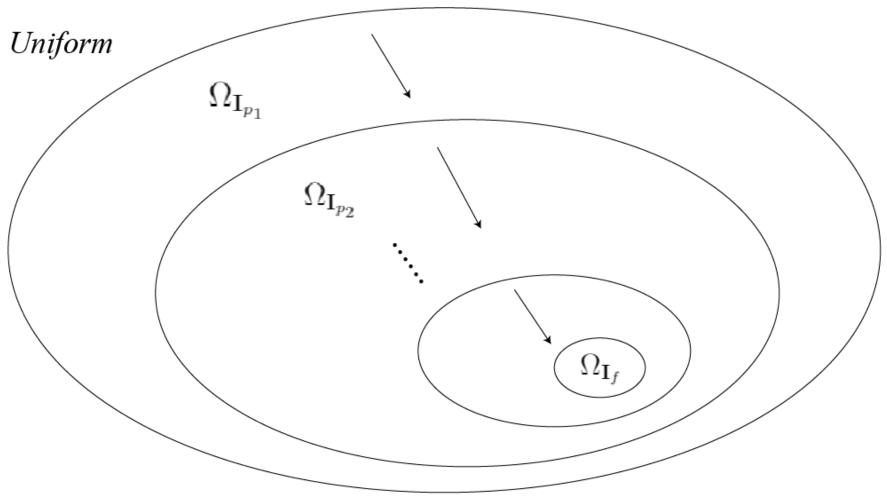


Figure 3.14: Reduction of constrained set of distributions from those sampled from uniform distribution, which corresponds to the entire pixel space, to an equivalence class of images that match projected statistics as the true samples. Here $\Omega_{\mathbf{I}_{p_1}}$ represent the set of images drawn from p_1, which follows Gibbs distribution by Maximum Entropy Principle. We pursue until images are close to images from $f(\mathbf{I})$, the true distribution.

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 Figure 3.14.

The figure can be interpreted as follows. We start from 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 Ω_{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 sheep herding. At first, the sheep are wandering in an infinitely large. At a certain time step, shepherds starts hurrying the sheep into a newly built fence, which defines a constrained space. Not wanting to be constrained further, the sheep push back against the fence, trying to escape. Shepherds then build a new, smaller fence inside the last one and hurrying the sheep further inside, and sheep further push back. This process is iterated until a shepherds stop building more fences.

Figure 3.15 gives 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 matches 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 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 FRAME model. We first fix a set of filters and try to learn Λ.

To re-iterate our definition, let image $\mathbf{I}$ be defined on a discrete domain $\mathcal{D}$, $\mathcal{D}$ 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

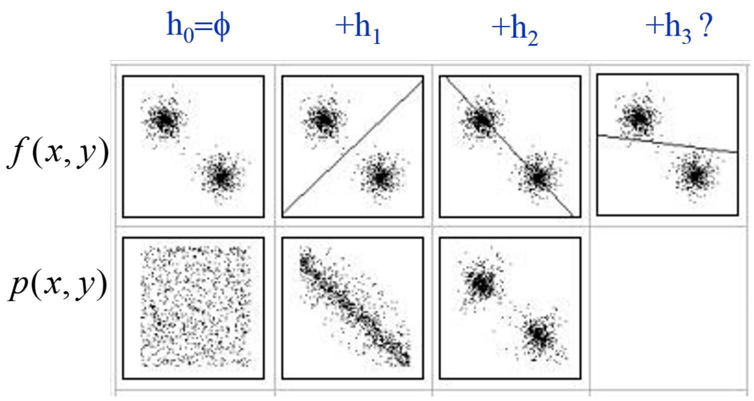


Figure 3.15: A toy example pursuit process as described by FRAME.

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 Function

Given an image $\mathbf{I}$ and a filter $F^{(\alpha)}$ with $\alpha = 1, 2, ..., K$ being an index of 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 $\delta(\cdot)$ 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

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

$$= \int e^{-2\pi iz} \, f^{(\xi)}(z) dz$$

where $\langle \cdot, \cdot \rangle$ is the inner product, and 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 big as image $\mathbf{I}$.

Our second motivation comes from psychophysical research on human texture perception, and the latter suggests that two homogeneous textures are often difficult to discriminate when they produce similar *marginal distributions* for responses from *a bank of filters* [12]. 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. 1) We limit our model to homogeneous textures, thus $f(\mathbf{I})$ is stationary with respect to location s. 2) 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 256×256 pixels, and the sizes of filters we used are limited to be less than 33×33 pixels. These filters can be linear or non-linear as we discussed in section (**??**). 3) Only a finite set of filters are used to estimate $f(\mathbf{I})$.

Assumptions 1) and 2) 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 assumption 1), $f_s^{(\alpha)}(z) = f^{(\alpha)}(z)$ is independent of s. By ergodicity, $H^{obs(\alpha)}(z)$ makes a consistent estimator of $f^{(\alpha)}(z)$. Assumption 2) ensures that the image size is lager 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 $S_K = \{F^{(\alpha)}, \alpha = 1, ..., K\}$ be a finite set of well selected filters, and $f^{(\alpha)}(z), \alpha = 1, ..., K$ are the corresponding marginal distributions of $f(\mathbf{I})$. We denote the probability distribution $p(\mathbf{I})$ which matches these marginal distributions as a set

$$\Omega_{p_K} = \{p(\mathbf{I}) \mid E_p[\delta(z - \mathbf{I}_s^{(\alpha)})] = f^{(\alpha)}(z) \ \ \forall z \in R, \ \forall \alpha = 1, ..., K, \ \forall s \in \mathcal{D}\}, \tag{3.13}$$

where $E_p[\delta(z - \mathbf{I}_s^{(\alpha)})]$ is the marginal distribution of $p(\mathbf{I})$ with respect to filter $F^{(\alpha)}$ at location s. Thus according to assumption 3), any $p(\mathbf{I}) \in \Omega_{p_K}$ is perceptually a good enough model for the texture, provided that we have enough well selected filters. Then we choose from Ω_{p_K} a distribution $p(\mathbf{I})$ which has the maximum entropy,

$$p(\mathbf{I}) = \arg\max\{-\int p(\mathbf{I}) \log p(\mathbf{I}) d\mathbf{I}\}, \tag{3.14}$$

$$\text{subject to} \qquad E_p[\delta(z - \mathbf{I}_s^{(\alpha)})] = f^{(\alpha)}(z), \quad \forall z \in R, \ \forall \alpha = 1, ..., K, \ \forall s \in \mathcal{D},$$

$$\text{and} \qquad \int p(\mathbf{I}) d\mathbf{I} = 1.$$

The reason for us to choose the maximum entropy distribution is that while $p(\mathbf{I})$ satisfies the constraints along some dimensions, it is made as random as possible in other unconstrained dimensions, since entropy is a measure of randomness. In other words, $p(\mathbf{I})$ should represent information no more than that is available.

Therefore an ME distribution gives the simplest explanation for the constraints and thus the purest fusion of the extracted features.

The constraints on equation (3.14) differ from the ones given in section (**??**) in that z takes continuous real values, hence there is an uncountable number of constraints, therefore, the Lagrange parameter λ takes the form as a function of z. Also since the constraints are the same for all locations $s \in \mathcal{D}$, λ should be independent of s. Solving this maximization problem gives the ME distribution:

$$\begin{aligned} p(\mathbf{I}; \Lambda_K, S_K) &= \frac{1}{Z(\Lambda_K)} \exp\{-\sum_s \sum_{\alpha=1}^{K} \int \lambda^{(\alpha)}(z)\delta(z - \mathbf{I}_s^{\alpha})dz\} && (3.15) \\ &= \frac{1}{Z(\Lambda_K)} \exp\{-\sum_s \sum_{\alpha=1}^{K} \lambda^{(\alpha)}(\mathbf{I}_s^{(\alpha)})\}, && (3.16) \end{aligned}$$

where $S_K = \{F^{(1)}, F^{(2)}, ..., F^{(K)}\}$ is a set of selected filters, and $\Lambda_K = (\lambda^{(1)}(\cdot), \lambda^{(2)}(\cdot), ..., \lambda^{(K)}(\cdot))$ is the Lagrange parameter. Note that in equation (3.16), 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 (3.16). Assume that the filter response $\mathbf{I}_s^{(\alpha)}$ is quantized into L discrete gray levels, therefore z takes values from set $\{z_1^{(\alpha)}, z_2^{(\alpha)}, ..., z_L^{(\alpha)}\}$. 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 piecewisely constant functions of L bins, and we denote these piecewise functions as vectors. $H^{(\alpha)} = (H_1^{(\alpha)}, H_2^{(\alpha)}, ..., H_L^{(\alpha)})$ 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 $\lambda^{(\alpha)} = (\lambda_1^{(\alpha)}, \lambda_2^{(\alpha)}, ..., \lambda_L^{(\alpha)})$.

So equation (3.15) is rewritten as:

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

by changing the order of summations:

$$\begin{aligned} p(\mathbf{I}; \Lambda_K, S_K) &= \frac{1}{Z(\Lambda_K)} \exp\{-\sum_{\alpha=1}^{K} \sum_{i=1}^{L} \lambda_i^{(\alpha)} H_i^{(\alpha)}\} \\ &= \frac{1}{Z(\Lambda_K)} \exp\{-\sum_{\alpha=1}^{K} \langle \lambda^{(\alpha)},\ H^{(\alpha)} \rangle\}. && (3.17) \end{aligned}$$

The virtue of equation (3.17) 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 $\Lambda_K = (\lambda^{(1)}, \lambda^{(2)}, ..., \lambda^{(K)})$ and S_K.
- Given an image $\mathbf{I}$, its histograms $H^{(1)}, H^{(2)}, ..., H^{(K)}$ are sufficient statistics, i.e. $p(\mathbf{I}; \Lambda_K, S_K)$ is a function of $(H^{(1)}, H^{(2)}, ..., H^{(K)})$.

We plug equation (3.17) into the constraints of the ME distribution, and solve for $\lambda^{(\alpha)}, \alpha = 1, 2, ..., K$ iteratively by the following equations,

$$\frac{d\lambda^{(\alpha)}}{dt} = E_{p(\mathbf{I};\Lambda_K, S_K)}[H^{(\alpha)}] - H^{obs(\alpha)}. \qquad (3.18)$$

In equation (**??**), 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 Λ_K. Equation (**??**) converges to the unique solution at $\Lambda_K = \hat{\Lambda}_K$ as we discussed in section (**??**), 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),

$$\begin{aligned}\hat{\Lambda}_K &= \arg\max_{\Lambda_K} \log p(\mathbf{I}^{obs};\Lambda_K,S_K) \\ &= \arg\max_{\Lambda_K} -\log Z(\Lambda_K) - \sum_{\alpha=1}^{K}\langle \lambda^{(\alpha)},\ H^{obs(\alpha)}\rangle. \end{aligned} \tag{3.19}$$

By gradient ascent, maximizing the log-likelihood gives equation (3.18), following property i) of the partition function $Z(\Lambda_K)$. In Equation (3.18), at each step, given Λ_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, for example, 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 3.4.2 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 $S_K = \{F^{(1)}, F^{(2)}, ..., F^{(K)}\}$.
Compute $\{H^{obs(\alpha)}, \quad \alpha = 1, ..., K\}$.
Initialize $\lambda_i^{(\alpha)} \leftarrow 0, \quad i = 1,2,...,L, \quad \alpha = 1,2,...,K$.
Initialize $\mathbf{I}^{syn}$ as a uniform white noise texture. [†]
Repeat
  Calculate $H^{syn(\alpha)}$ $\alpha = 1,2,...,K$ from $\mathbf{I}^{syn}$, use it for $E_{p(\mathbf{I};\Lambda_K,S_K)}(H^{(\alpha)})$ .
  Update $\lambda^{(\alpha)}$ $\alpha = 1,2,...,K$ by equation (??), 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 $\frac{1}{2}\sum_i^L \mid H_i^{obs(\alpha)} - H_i^{syn(\alpha)} \mid \ \leq \epsilon$ for $\alpha = 1,2,...,K$.
```

Algorithm 1: The learning algorithm

In algorithm 3.4.2, to compute $p(\mathbf{I}_s = val \mid \mathbf{I}_{-s})$, we set $\mathbf{I}_s$ to val, due to Markov property, we only need to compute the changes of $\mathbf{I}^{(\alpha)}$ at the neighborhood of s. The size of the neighborhood is determined by the size of filter $F^{(\alpha)}$. With the updated $\mathbf{I}^{(\alpha)}$, we calculate $H^{(\alpha)}$, and the probability is normalized such that $\sum_{val=0}^{G-1} p(\mathbf{I}_s = val \mid \mathbf{I}_{-s}) = 1$.

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

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

Algorithm 2: The Gibbs Sampler for w sweeps

In 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 the filtering theory and the random fields models, and it interprets many previous methods for texture modeling in a unified view of point.

Filter Selection

After Λ 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 $\mathcal{B}$ is to search for all possible combinations of K filters in 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 $S_0 = \emptyset$ and hence $p(\mathbf{I}; \Lambda_0, S_0)$ a uniform distribution, and then sequentially introduce one filter at a time. Namely, we want to find one that reduces $KL(f, p(\mathbf{I}; \Lambda))$ the most (by minimum entropy principle).

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

$$p(\mathbf{I}; \Lambda_k, S_k) = \frac{1}{Z(\Lambda_k)} \exp\{-\sum_{\alpha=1}^{k} \langle \lambda^{(\alpha)},\ H^{(\alpha)} \rangle\}, \tag{3.20}$$

so that $E_{p(\mathbf{I};\Lambda_k,S_k)}[H^{(\alpha)}] = f^{(\alpha)}$ for $\alpha = 1, 2, ..., k$. 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.,

$$F^{k+1} = \arg\max_{F^{(\beta)} \in \mathcal{B}/S_k} D(E_{p(\mathbf{I};\Lambda_k,S_k)}[H^{(\beta)}], f^{(\beta)}). \tag{3.21}$$

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

$$D(E_{p(\mathbf{I};\Lambda_{k+1},S_{k+1})}[H^{(\beta)}], f^{(\beta)}) = KL(f, p(\mathbf{I}; \Lambda_k, S_k)) - KL(f, p(\mathbf{I}; \Lambda_{k+1,S_{k+1}})). \tag{3.22}$$

Note that the above difference is a distance measure since

$$KL(f, p(\mathbf{I}; \Lambda_k, S_k)) - KL(f, p(\mathbf{I}; \Lambda_{k+1}, S_{k+1}))) = KL(p(\mathbf{I}; \Lambda_{k+1}, S_{k+1})), p(\mathbf{I}; \Lambda_k, S_k)) \tag{3.23}$$

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

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_+,

$$\begin{aligned} KL(f,p) - KL(f,p_+) &= \textit{entropy}(f) - \mathrm{E}_f[\log p] - \textit{entropy}(f) + \mathrm{E}_f[\log p_+] \\ &= \mathrm{E}_f[\log p_+] - \mathrm{E}_f[\log p] \end{aligned}$$

We refer back to Theorem **??** and it becomes clear that $\mathrm{E}_f[\log p] = -\textit{entropy}(p)$ and $\mathrm{E}_f[\log p_+] = -\textit{entropy}(p_+)$. Then $\mathrm{E}_f[\log p_+] - \mathrm{E}_f[\log p] = \textit{entropy}(p_+) - \textit{entropy}(p)$, which is the definition of information gain. □

Information gain, however, is not directly computable. We then need to approximate it with Taylor expansion.

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 detailed reason, please refer to Chapter 8). As λ^{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 $D(h)$. □

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 bits reduction, i.e.,

$$F^{k+1} = \underset{F^{(\beta)} \in \mathcal{B}/S_k}{\arg\max} \frac{1}{2}|f^{(\beta)} - E_{p(\mathbf{I};\Lambda_k,S_k)}[H^{(\beta)}]|_p. \tag{3.24}$$

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.,

$$F^{k+1} = \underset{F^{(\beta)} \in \mathcal{B}/S_k}{\arg\max} \frac{1}{2}|H^{obs(\beta)} - H^{syn(\beta)}|. \tag{3.25}$$

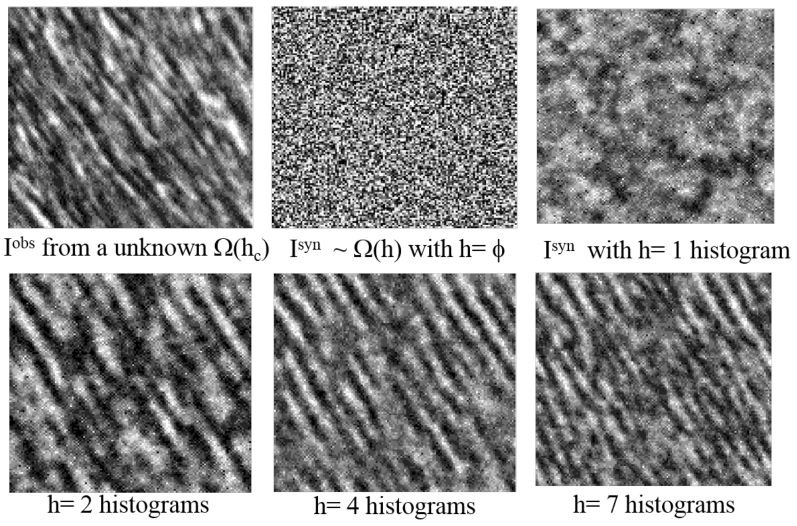


Figure 3.16: Synthesis of a fur texture. a is the observed texture, b,c,d,e,f are the synthesized textures using $K = 0, 1, 2, 3, 6$ filters respectively.

The filter selection procedure stops when the $d(\beta)$ for all filters $F^{(\beta)}$ in the filter bank are smaller than a constant ϵ. 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 ϵ.

Finally, we propose algorithm 3.4.2 for filter selection.

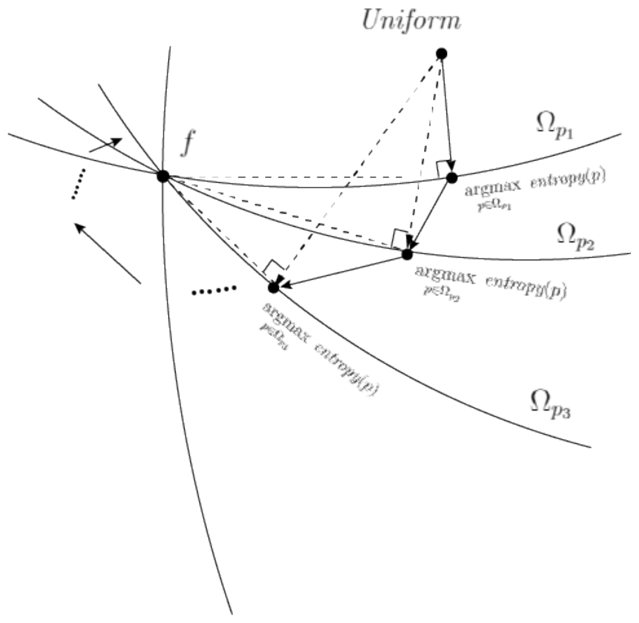
Figure (3.16.a) is the observed image of animal fur. We start from the uniform noise image in figure (3.16.b).

The first filter picked by the algorithm is a Laplacian of Gaussian filter $LG(1.0)$ and its window size is 5×5. It has the largest error $d(\beta) = 0.611$ among all the filters in the filters bank. Then we synthesize texture as shown in figure (3.16.c), which has almost the same histogram at the subband 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 Figure 3.17. 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 find the distribution on this set closest to random uniform (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. Next 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 notice that Ω_{p_2} has lower dimension than Ω_{p_1} and every set after Ω_{p_2} has lower dimension due to additional constraints. True distribution f again is also in this set as it satisfies the additional feature statistic. In this new constrained set, we start again from random uniform 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 8.

```
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 $\alpha = 1, ..., |\mathcal{B}|$ 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 $\mathbf{I}^{syn(\beta)}$
      $d(\beta) = \frac{1}{2} \mid H^{obs(\beta)} - H^{syn(\beta)} \mid$, ‡
    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



Uniform
f
Ω_{p_1}
Ω_{p_2}
Ω_{p_3}
$\arg\max_{p \in \Omega_{p_1}} entropy(p)$
$\arg\max_{p \in \Omega_{p_2}} entropy(p)$
$\arg\max_{p \in \Omega_{p_3}} entropy(p)$

Figure 3.17: A graphical view of the FRAME learning algorithm in distribution space.

3.5 The Texture Ensemble

The following section presents theoretical work on texture theories and establishes connections between FRAME and statistical physics and Julesz Ensemble. This section is highly technical and it is encouraged for interested readers to go through the original works [203] [202].

3.5.1 Ensembles in Statistical Physics

We have developed the FRAME model based on the maximum entropy principle and there is a simpler and more profound way to derive or justify the 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 [17]). Fig 3.18 illustrates three types of physical systems which are interesting to us.

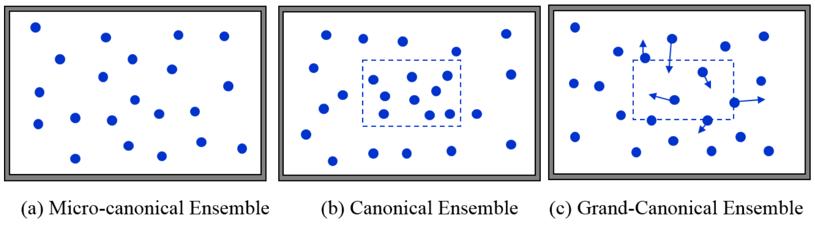

(a) Micro-canonical Ensemble (b) Canonical Ensemble (c) Grand-Canonical Ensemble

Figure 3.18: Three types of ensembles in statistical physics.

1. Micro-canonical ensembles. Fig. 3.18.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 is called 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 purpose. Obviously Ω_{mce} is a deterministic set or an equivalence class for all states that satisfy a 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. 3.18.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,

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

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."

theorem states that the Gibbs model p_{Gib} is a conditional probability of the uniform model p_{unif}. This conclusion is extremely important because it bridges a deterministic set Ω_{mce} with a descriptive model p_{Gib}. We consider this is an origin of probability for 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. 3.18.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 β_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.

3.5.2 Texture Ensemble

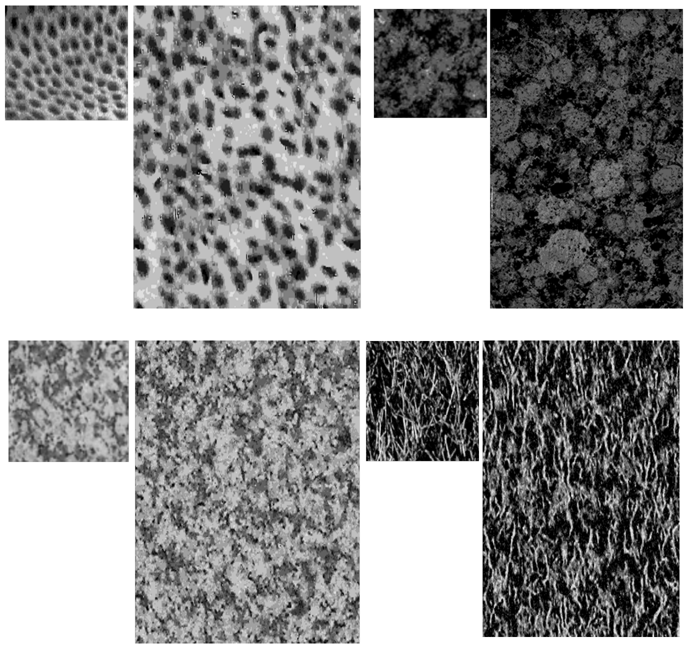


Figure 3.19: For each pair of texture images, the image on the left is the observed image, and the image on the right is the image randomly sampled from the Julesz texture ensemble.

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

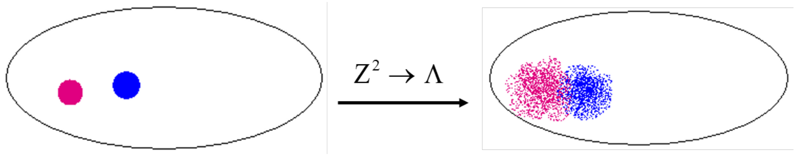


Figure 3.20: Julesz ensembles of different textures are disjoint in the space of infinite images. They will overlap in the space of image patches of finite size.

some bins of joint statistics may also be important as long as they can be reliably estimated from finite observations.

Given K Gabor filters as feature detectors $\{F^{(1)}, ..., F^{(K)}\}$, we convolve the filters with image $\mathbf{I}$ to obtain the subband filtered images $\{\mathbf{I}^{(1)}, ..., \mathbf{I}^{(K)}\}$, 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 subband images,

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

We use $\mathbf{H}(\mathbf{I}) = (H^{(1)}(\mathbf{I}), ..., H^{(K)}(\mathbf{I}))$ to denote the unnormalized histograms. We assume that boundary conditions are properly handled (e.g., periodic boundary condition). It should be noted that the conclusions of this paper 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 Ω_D into equivalence classes $\Omega_D(\mathbf{h}) = \{\mathbf{I} : \mathbf{h}(\mathbf{I}) = \mathbf{h}\}$, as we did for the iid case. 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

$$q(\mathbf{I}; \mathcal{H}) = \begin{cases} 1/|\Omega_D(\mathcal{H})|, & \text{if } \mathbf{I} \in \Omega_D(\mathcal{H}), \\ 0, & \text{otherwise}, \end{cases} \tag{3.26}$$

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* $\mathbf{h}(\mathbf{I}) = (h^{(1)}(\mathbf{I}), ..., h^{(K)}(\mathbf{I}))$*, a Julesz ensemble of type* $\mathbf{h}$ *is a limit of* $q(\mathbf{I}; \mathcal{H})$ *as* $D \to Z^2$ *and* $\mathcal{H} \to \mathbf{h}$ *with some boundary condition.*[§]

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* $\mathbf{h}(\mathbf{I})$.

[§]We assume $D \to Z^2$ in the sense of van Hove, i.e., the ratio between the boundary and the size of D goes to 0.

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 $D \subset \mathrm{Z}^2$, and the intensity at pixel $v \in D$ is denoted by $\mathbf{I}(v) \in \mathcal{G} = \{1, 2, ..., g\}$. 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 $p = (p_1, ..., p_g)$ where $\sum_i p_i = 1$. The distribution of $\mathbf{I}$ can be written as a FRAME model

$$p(\mathbf{I}; \beta) = \prod_{v \in D} p_{\mathbf{I}(v)} = \prod_{i=1}^{g} p_i^{H_i(\mathbf{I})} = \exp\{\langle \log p, H(\mathbf{I})\rangle\} = \exp\{\langle \beta, H(\mathbf{I})\rangle\}, \tag{3.27}$$

where $H(\mathbf{I}) = (H_1(\mathbf{I}), ..., H_g(\mathbf{I}))$ is the unnormalized intensity histogram of $\mathbf{I}$, i.e., H_i is the number of pixels whose intensities equal to i. $\beta = (\log p_1, ..., \log p_g)$ 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 $\mathbf{I}$.

3) Equivalent class. Let $\Omega_D(h)$ be the set of images with $h(\mathbf{I}) = h$[¶], 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 figure 3.21, each equivalence class $\Omega_D(h)$ is mapped into one type h on a *simplex* – a plane defined by $h_1 + \cdots + h_g = 1$ and $h_i \geq 0, \forall i$ in an g-dimensional space.

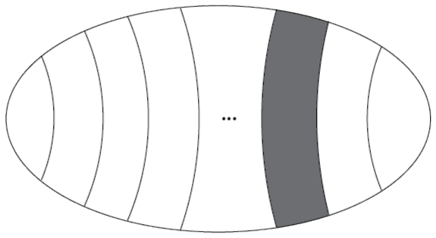


Figure 3.21: The partition of image space into equivalence class, and each class corresponds to an h on the probability simplex.

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 $D \to \mathrm{Z}^2$, where statistical fluctuations vanish. Therefore, we may attempt to define the Julesz ensemble as the limit of $\Omega_D(h)$ as $D \to \mathrm{Z}^2$, or even more directly, as the set of images $\mathbf{I}$ defined on 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 each equivalence class $\Omega_D(h)$ a probability

[¶]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.

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 $D \to \mathrm{Z}^2$.

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 $\mathcal{H}$ goes to h as $D \to \infty$. 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 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

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

Thus for large enough lattice, the volume of $\Omega_D(h)$ is said to be in the order of 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

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

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 holds when $h = p$.

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

$$p(\Omega_D(h); \beta) \sim \exp^{-|D|\text{KL}(h||p)} \tag{3.28}$$

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

7) Typical vs. most likely images. Suppose among $p_1, ..., p_g$, 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 $D \to \mathrm{Z}^2$. 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 $n \to \infty$, the sum $e^{5n} + e^{3n}$ is dominated by e^{5n}, and the order of this sum is still 5, i.e., $\lim_{n\to\infty} \frac{1}{n} \log(e^{5n} + e^{3n}) = 5$. 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 (3.28), we have

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

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

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

That is, the total probability for $\Omega_D(\mathcal{H})$ has an exponential order $\mathrm{KL}(h^*||p)$, 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 $\mathrm{KL}(h^*||p) = 0$.

Clearly, as $|D| \to \infty$, 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 tight 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.[‖] We shall show that if we fix D_0 and let $D \to \mathrm{Z}^2$, then $q(\mathbf{I}_{D_0}; h)$ goes to the FRAME model (see equation (3.27)) 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

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

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

$$\begin{aligned}
q(\mathbf{I}_{D_0}; h) &= \frac{|\Omega_{D/D_0}(\frac{h|D|-H(\mathbf{I}_{D_0})}{|D/D_0|})|}{|\Omega_D(h)|} \\
&= \frac{(|D| - |D_0|)!/\prod_{i=1}^g (h_i|D| - H_i(\mathbf{I}_{D_0}))!}{|D|!/\prod_{i=1}^g (h_i|D|)!} \\
&= \frac{\prod_{i=1}^g (h_i|D|)(h_i|D| - 1)...(h_i|D| - H_i(\mathbf{I}_{D_0}) + 1)}{|D|(|D| - 1)...(|D| - |D_0| + 1)} \\
&= \frac{\prod_{i=1}^g h_i(h_i - 1/|D|)...(h_i - (H_i(\mathbf{I}_{D_0}) - 1)/|D|)}{(1 - 1/|D|)...(1 - (|D_0| - 1)/|D|)} \\
&\to \prod_{i=1}^g h_i^{H_i(\mathbf{I}_{D_0})} \text{ as } |D| \to \infty.
\end{aligned}$$

Therefore, the distribution of $\mathbf{I}_{D_0}$ is the FRAME model (see equation (3.27)) 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.

[‖]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.

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 traditional argument in statistical physics (e.g., Laudu, 1959). It is not as rigorous as modern treatments, but is much more revealing.

Suppose the feature statistics is $\mathbf{h}(\mathbf{I})$ 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 associate 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 $D \to \mathrm{Z}^2$. We assume that D_0 is sufficiently smaller than D_1 so that the neighborhood of D_0, ∂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

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

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,

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

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 $\mathcal{H} \to \mathbf{h}$ as $D \to \mathrm{Z}^2$. Let $\Omega_D(\mathcal{H}) = \{\mathbf{I} : \mathbf{h}(\mathbf{I}) in \mathcal{H}\}$, then we have the following

Proposition 3. *The limit*

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

exists. Let $s(\mathbf{h}) = \lim_{\mathcal{H} \to \bar{\mathcal{H}}} s(\mathcal{H})$*, 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 $\boldsymbol{\beta} = s'(\mathbf{h})$, then, as $D \to \mathrm{Z}^2$, $\mathbf{h}_- \to \mathbf{h}$, and $s'(\mathbf{h}_-) \to \boldsymbol{\beta}$. Therefore,

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

so

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

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 $\boldsymbol{\beta}$ can be identified as the derivative of the entropy function $s(\mathbf{h})$, $\boldsymbol{\beta} = s'(\mathbf{h})$.

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 $D \to \mathrm{Z}^2$.

Consider the FRAME model

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

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

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

The asymptotic behavior of this probability is

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

For the last term, we have

Proposition 4. *The limit*

$$\rho(\boldsymbol{\beta}) = \lim_{D \to \mathbf{Z}^2} \frac{1}{|D|} \log Z_D(\boldsymbol{\beta})$$

exists and is independent of the boundary condition. $\rho(\boldsymbol{\beta})$ 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 $p(\mathbf{I}; \boldsymbol{\beta})$ is*

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

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 $\boldsymbol{\beta}$ (otherwise, the probability will go unbounded). This $s_\beta(\mathbf{h})$ can be identified with $-\mathrm{KL}(h||p)$ in the simple iid case.

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

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

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

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 3.18.

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 by the same approach as we did.

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 [188] 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 [137]. 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

$$\begin{cases} \frac{\partial A}{\partial t} = D_a \Delta A + R_a(A, B) \\ \frac{\partial B}{\partial t} = D_b \Delta B + R_b(A, B) \end{cases} \tag{3.29}$$

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) [68]. 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 **??**, 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 $\mathbf{I}(x, y, t)$ [148]. Similar to heat diffusion equation, their equation also simulate the "heat" diffusion process,

$$\frac{d\mathbf{I}}{dt} = \text{div}(c(x, y, t)\nabla \mathbf{I}), \quad \mathbf{I}(x, y, 0) = \mathbf{I}_0 \tag{3.30}$$

where $\nabla \mathbf{I} = (\frac{\partial \mathbf{I}}{\partial x}, \frac{\partial \mathbf{I}}{\partial y})$ is the intensity gradient and div is the divergence operator, $\text{div}(\vec{V}) = \nabla_x P + \nabla_y Q$ 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

$$\frac{d\mathbf{I}}{dt} = \nabla_x (\frac{1}{1 + (\nabla_x \mathbf{I}/b)^2} \nabla_x \mathbf{I}) + \nabla_y (\frac{1}{1 + (\nabla_y \mathbf{I}/b)^2} \nabla_y \mathbf{I}) \tag{3.31}$$

where b is a scaling constant. It is easy to see that equation 3.31 minimizes the following energy function by gradient descent, just as in heat diffusion process in the section above,

$$U(\mathbf{I}) = \int \int \lambda(\nabla_x \mathbf{I}) + \lambda(\nabla_y \mathbf{I}) dx dy \tag{3.32}$$

where $\lambda(\xi) = \textit{const.log}(1 + (\xi/b)^2)$ and $\lambda'(\xi) = \textit{const.}\frac{\xi}{1+(\xi/b)^2}$ are plotted in figure **??**

Although the anisotropic diffusion equations can be adopted for removing noise and enhancing edges [143], $I(z, y, t)$ converges to a flat image as $t \to \infty$ in the Perona-Malik equation.

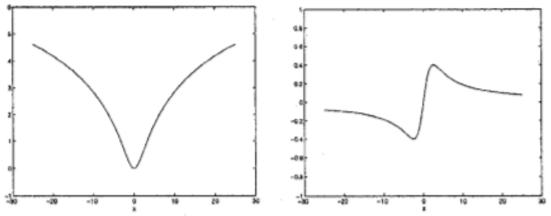


Figure 3.22: On the left is $\lambda(\xi) = \mathit{const.} \log\left(1 + (\xi/b)^2\right)$ and on the right is $\lambda'(\xi) = \mathit{const.} \frac{\xi}{1+(\xi/b)^2}$

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 as we did.

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

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

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

$$U(\mathbf{I}; \Lambda, S) = \sum_{i=1}^{n} \sum_{(x,y)\in D} \lambda_i(F_i * \mathbf{I}(x, y)) \tag{3.33}$$

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 paper, we shall only study the following linear filters.

1. An intensity filter $\delta(\cdot)$, and gradient filters ∇_x, ∇_y.

2. The Laplacian of Gaussian filters.

$$LoG(x, y, s) = \mathit{const.}(x^2 + y^2 - s^2)e^{-\frac{x^2+y^2}{s^2}} \tag{3.34}$$

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 3×3 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.

$$G(x, y, s, \theta) = \mathit{const.} \cdot \mathit{Rot}(\theta) \cdot e^{\frac{1}{2s^2}(4x^2+y^2)} e^{-i\frac{2\pi}{s}x} \tag{3.35}$$

It is a sine wave at frequency $\frac{2\pi}{s}$ modulated by an elongated Gaussian Function, and rotated at angle θ. We denote the real and image 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 3.33 in discretized bins using FRAME, we found that texture patterns generally exhibit two families of functions similar to reaction-diffusion in

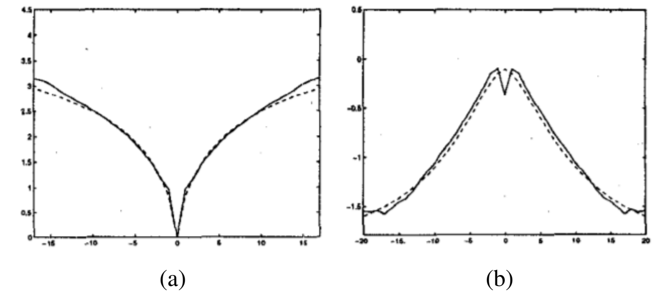

(a) (b)

Figure 3.23: This figure shows two classes of functions. (a) shows diffusion functions. b) shows reaction functions. Dotted lines show the fitted ϕ and ψ functions, respectively.

chemical processes as shown in Figure 3.23. We use the following two families of functions to fit our discretized findings, and the fitted curves are shown as dashed lines in Figure 3.23.

$$\phi(\xi) = a(1 - \frac{1}{1 + (|\xi - \xi_0|/b)^\gamma}), \quad a > 0 \tag{3.36}$$

$$\psi(\xi) = a(1 - \frac{1}{1 + (|\xi - \xi_0|/b)^\gamma}), \quad a < 0 \tag{3.37}$$

where ξ_0, b are respectively the translation and scaling constants, and $\|a\|$ weights the contribution of the filter. $\phi(\xi)$, the diffusion function, assigns lowest energy (or highest probability) to filter responses around ξ_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.

Now we can design our potential function in equation 3.33 to be

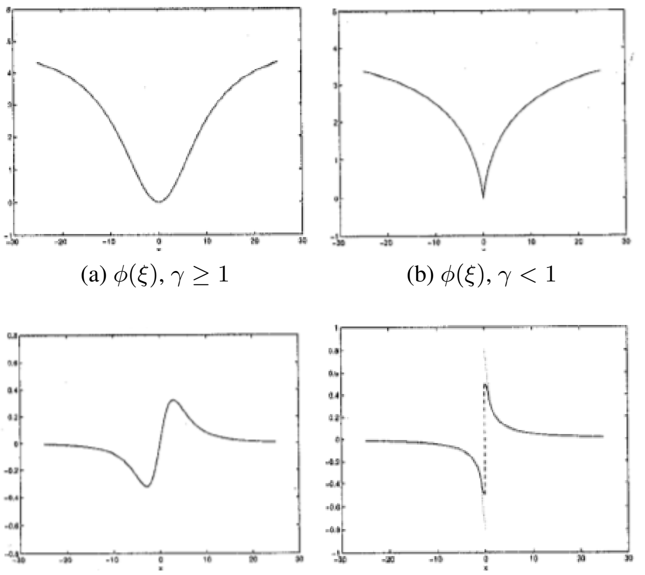
$$U(\mathbf{I}; \Lambda, S) = \sum_{i=1}^{n_d} \sum_{(x,y)\in D} \phi_i(F_i * \mathbf{I}(x,y)) + \sum_{i=n_d}^{n} \sum_{(x,y)\in D} \phi_i(F_i * \mathbf{I}(x,y)) \tag{3.38}$$

Note that the filter set is divided into two parts $S = S_d \cup S_r$, with $S_d = \{F_i,\, i = 1, 2, ..., n_d\}$ and $S_r = \{F_i,\, i = n_d + 1, ..., n\}$. 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:

$$\frac{d\mathbf{I}}{dt} = \sum_{i=1}^{n_d} F_i^- * \phi_i'(F_i * \mathbf{I}) + \sum_{i=n_d}^{n} F_i^- * \psi_i'(F_i * \mathbf{I}) \tag{3.39}$$

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 as they favor large filter responses. We call equation 3.39 the Gibbs Diffusion And Reaction Equation (GRADE).



(a) $\phi(\xi), \gamma \geq 1$ (b) $\phi(\xi), \gamma < 1$

(c) $\phi'(\xi), \gamma \geq 1$ (d) $\phi'(\xi), \gamma < 1$

Figure 3.24: Here $\phi(\xi) = a(1 - \frac{1}{1+(|\xi-\xi_0|/b)^\gamma}),\ \ a > 0$, and its derivative $\phi'(\xi)$ for a), b) is $\gamma = 2.0$ and b), d)$\gamma = 0.8$.

3.6.5 Properties of GRADE

Property 1: A General Statistical Framework

GRADE as in equation 3.39 can be considered as an extension to heat diffusion equation as in 3.30 on a discrete lattice by defining a vector field

$$\vec{V} = (\phi_1'(\cdot), ..., \phi_{n_d}'(\cdot), \psi_{n_d+1}'(\cdot), ..., \psi_n'(\cdot))$$

and divergence operator can be generalized to

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

Thus equation 3.39 can be written as

$$\frac{d\mathbf{I}}{dt} = \text{div}(\vec{V}) \tag{3.40}$$

Compared to equation 3.30 which transfers "heat" among adjacent pixels, equation 3.40 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 div $= \nabla_x + \nabla_y$, thus equation 3.30 and 3.31 are just special cases of equation 3.40.

Property 2: Diffusion

Figure 3.24a and 3.24b shows two best-fit potential functions. Figure 3.24a shows a round tip at $\xi = 0$, and Figure 3.24b 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

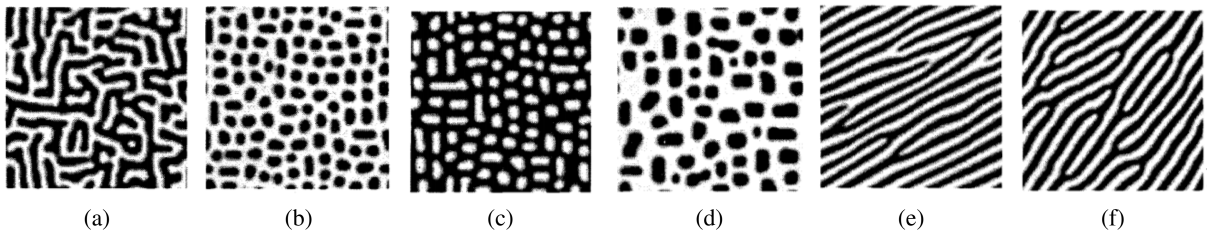


Figure 3.25: Generated texture patterns.

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.

Property 3: Reaction

The other class of potential function is reaction function. We refer back to Figure 3.23 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

$$d\mathbf{I} = -\nabla U(\mathbf{I})dt + \sqrt{2T(t)}dw_t \tag{3.41}$$

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 - Laplacian and 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 [96]). 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 Julesz's Confusion between Textures and Textons

In *psychophysics*, Julesz [96] 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.

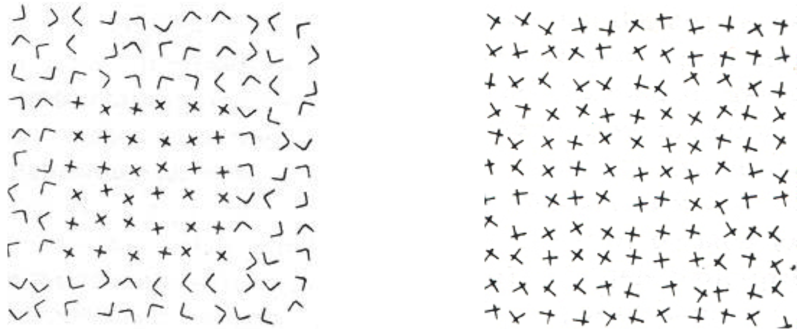


Figure 4.1: Pre-attentive vision is sensitive to local patterns called textons.

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 left pair is instantaneous ($100 - 200\ ms$) and independent of the number of distractors. In contrast, for the right 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 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

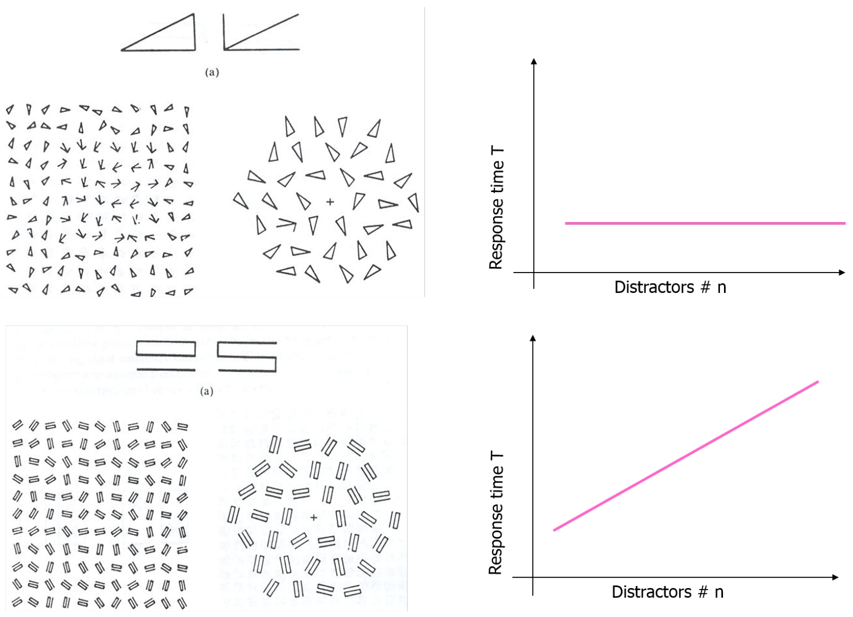


Figure 4.2: Julesz textures show that pre-attentive vision are sensitive to local image structures such as edges, bars, and end points.

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 interwaven 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 implicity 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.

4.2 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 256×256 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 [7].

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 [86]. 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 fourth 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 [144] who learned some over-complete image bases from natural image patches (12×12 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 as a linear superposition of some image bases which minimizes some measure of dependence between the coefficients of these bases [10].

4.2.2 Olshausen-Field model

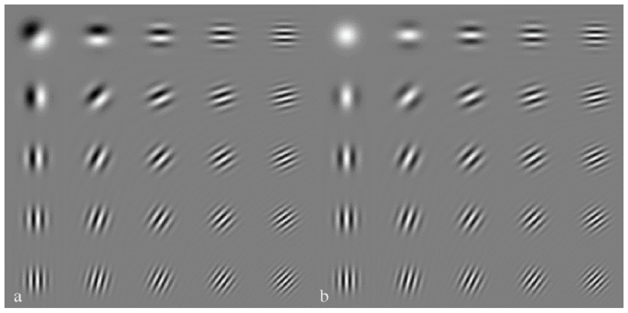


Figure 4.3: Gabor wavelets are sine and cosine waves multiplied by Gaussian functions.

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

$$\mathbf{\Psi} = \{\boldsymbol{\psi}_\ell(u, v), \ell = 1, ..., L_\psi\}.$$

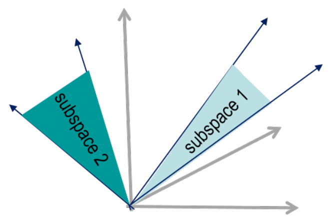


Figure 4.4: The sparse coding model assumes that the observed signals lie on the low dimensional subspaces spanned by the basis vectors.

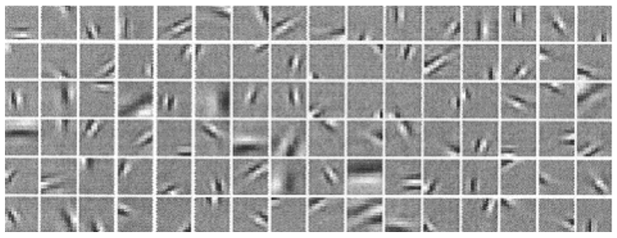


Figure 4.5: The above image patches are the basis vectors learned from natural image patches by the sparse coding model.

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

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

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 Δ plus a Gaussian noise image $\mathbf{n}$.

$$\mathbf{I} = \sum_{i}^{n_B} \alpha_i \cdot \boldsymbol{\psi}_i + \mathbf{n}, \qquad \boldsymbol{\psi}_i \in \Delta,\ \forall i, \tag{4.1}$$

where n_B is the number of bases and α_i is the *coefficient* of the i-th base $\boldsymbol{\psi}_i$.

As Δ is over-complete[*], the variables $(\ell_i, \alpha_i, x_i, y_i, \tau_i, \sigma_i)$ indexing a base ψ_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*,

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

If we view each base $\boldsymbol{\psi}_i$ 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

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

[*]The number of bases in Δ is often 100 times larger than the number of pixels in an image.

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

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 [144].[†] For example, $p(\alpha)$ is chosen to be a Laplacian distribution, or a mixture of two Gaussians with σ_1 close to zero. For all

$i = 1, ..., n_B$,

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

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 σ_1 close to zero,

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

A slight confusion in the literature 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.

In the above image model, the base map $\mathbf{B}$ includes the hidden variables and the dictionary $\mathbf{\Psi}$ are parameters. For example, Olshausen and Field used $L_\psi = 144$ base functions, each being a 12×12 matrix. Following an EM-learning algorithm, they learned $\mathbf{\Psi}$ from a large number of natural image patches. Fig. 11.5 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 12×12 pixel patches, therefore they didn't really inferred the hidden variables for the transformation A_i. Thus the learned bases are not aligned at centers and are rather noisy.

4.2.3 A three-level generative model

Our comparison study leads us 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 $\mathbf{\Psi}$ 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 $\mathbf{\Pi}$ with some orthogonal transforms. Each texton element in $\mathbf{T}$ consists of a few bases with a deformable geometric configuration. So we have,

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

with

$$\mathbf{\Psi} = \{\boldsymbol{\psi}_\ell, \ell = 1, 2, ..., L_\psi\}, \quad \text{and} \quad \mathbf{\Pi} = \{\boldsymbol{\pi}_\ell; \ell = 1, 2, ..., \mathbf{L}_\pi\}.$$

[†]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.

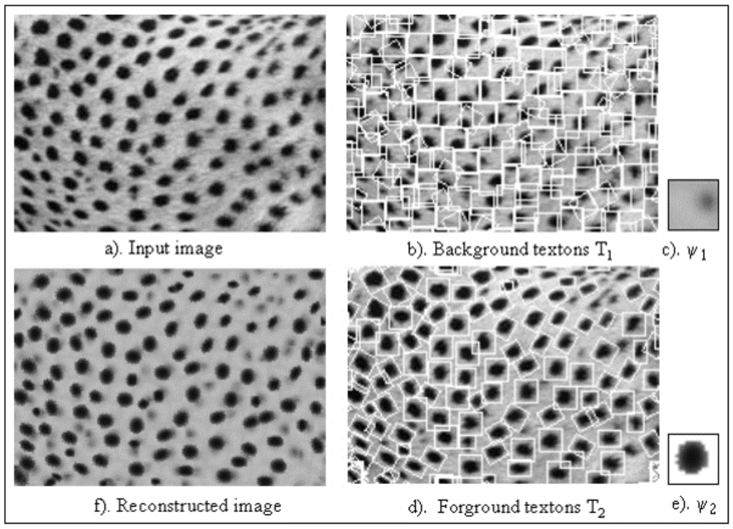


Figure 4.6: The observed image can be reconstructed from foreground and background textons.

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-occurence of the image bases.

To clarify terminology, a base function $\psi \in \mathbf{\Psi}$ 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 $\pi \in \mathbf{\Pi}$ 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 $\mathbf{\Pi}$.

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

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

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 paper 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. igure?, we first show the three base functions in $\mathbf{\Psi}$ (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 α_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 α_i. We put these secondary bases in another sub-layer (see the second column of Fig. When these image bases are superpositioned, they generate a reconstructed image (see the third column in Fig.

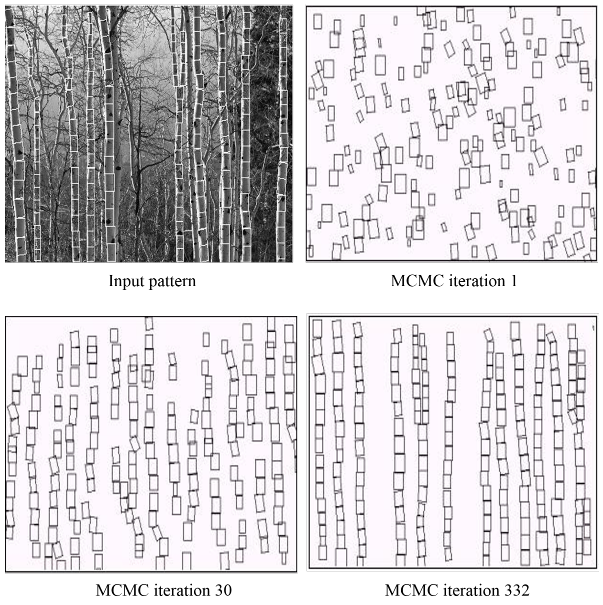


Figure 4.7: The spatial arrangement of textons can be modeled by a point process model.

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 $\Theta = (\mathbf{\Psi}, \mathbf{\Pi}, \boldsymbol{\kappa})$.

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

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

We rewrite the base map as

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

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

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

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

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

Bases in ϖ_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

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

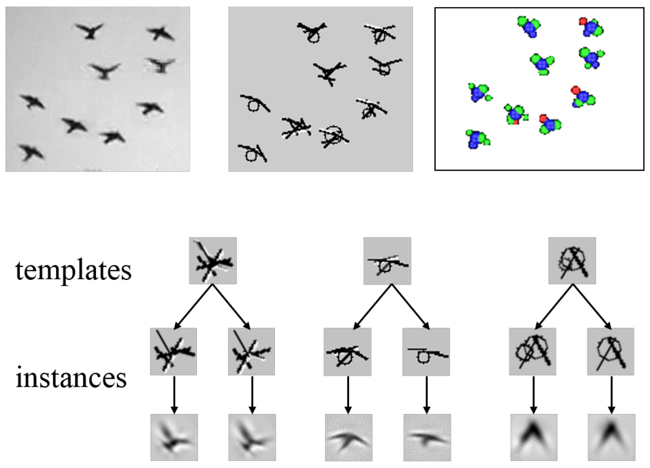


Figure 4.8: A texton template is a deformable composition of basis vectors.

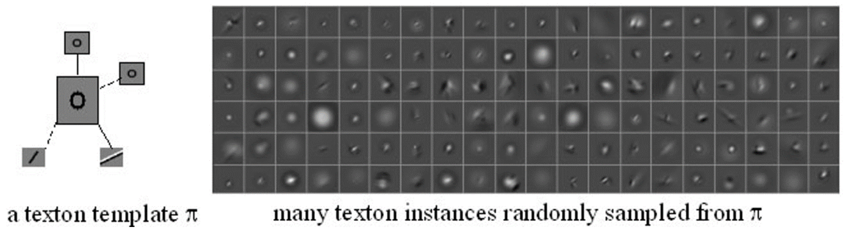


Figure 4.9: A texton template is a deformable composition of basis vectors.

Each texton element T_j is specified by its type ℓ_j, photometric contrast α_j, translation (x_j, y_j), rotation τ_j, scaling σ_j and deformation vector δ_j. A texton $\pi \in \mathbf{\Pi}$ consists of m image bases with a certain deformable configuration

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

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

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

$p(\mathbf{T};\boldsymbol{\kappa})$ 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 $\mathbf{I}^{\text{obs}}$,

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

In $p(\mathbf{I};\Theta)$ above, the parameters Θ (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.

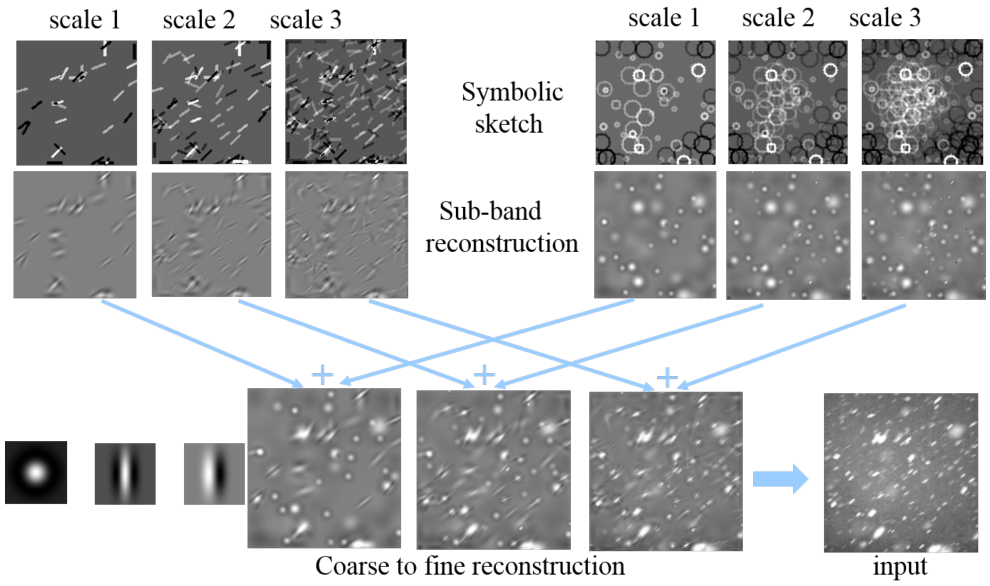


Figure 4.10: The image can be decomposed into sub-bands at multiple scales.

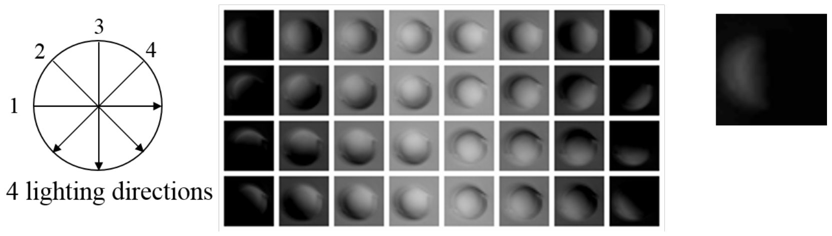


Figure 4.11: The concept of textons can also be generalize to incorporate lighting variations.

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

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

where ϵ is an approximation error which diminishes as sufficient data are available for training. In practice, ϵ may decide the complexity of the models, and thus the number of base functions L_ψ and textons L_π. 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. We 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

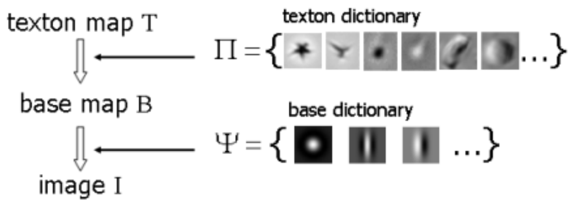


Figure 4.12: A three-level generative model: an image **I** is a linear addition of some image bases selected from a base dictionary Ψ, such as Gabor or Laplacian of Gaussians. The base map is further generated by a smaller number of textons selected from a texton directionry Π. Each texton consists of a number of bases in certain deformable configurations.

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" in this . 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. We present a sequence of experiments for learning the geometric, dynamic, and photometric structures from images and videos, and we also present some comparison studies with K-means clustering, sparse coding, independent component analysis, and transformed component analysis. We shall discuss how general textons can be learned from generic natural images.

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 $\{\mathbf{I}_m, m = 1, ..., M\}$ be a set of small image patches. For example, they might be 12×12 patches, in which case $\mathbf{I}_m \in \mathrm{R}^{12\times12}$. We may think of each $\mathbf{I}_m$ as a two-dimensional function defined on the 12×12 lattice. The Olshausen-Field model seeks to represent these images by

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

where $(B_i, i = 1, ..., N)$ is a dictionary of basis functions defined on the same image lattice (e.g., 12×12) 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 $(c_{m,i}, i = 1, ..., N)$ 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 α is the orientation. We call such a dictionary self-similar, and we call (x, s, α) the geometric attribute of $B_{x,s,\alpha}$.

Model (4.7) then becomes

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

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, α) are properly discretized (default setting: α 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 ℓ_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

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

where $n \ll N$ is a small number, and $(x_{m,i}, s_{m,i}, \alpha_{m,i}, i = 1, ..., n)$ are the geometric attributes of the selected basis functions whose coefficients $(c_{m,i})$ are non-zero. The attributes $(x_{m,i}, s_{m,i}, \alpha_{m,i}, i = 1, ..., n)$ 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 $\{\mathbf{I}_m, m = 1, ..., M\}$. 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 4.13 for 9 image patches of deer. We call such image patches aligned.

The active basis model is of the following form

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

where $\mathbf{B} = (B_{x_i,s,\alpha_i}, i = 1, ..., n)$ 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. $\mathbf{B}_m = (B_{x_i+\Delta x_{m,i},s,\alpha_i+\Delta\alpha_{m,i}}, i = 1, ..., n)$ 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 α_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 $\Delta\alpha_{m,i} \in \{-1, 0, 1\} \times \pi/16$).

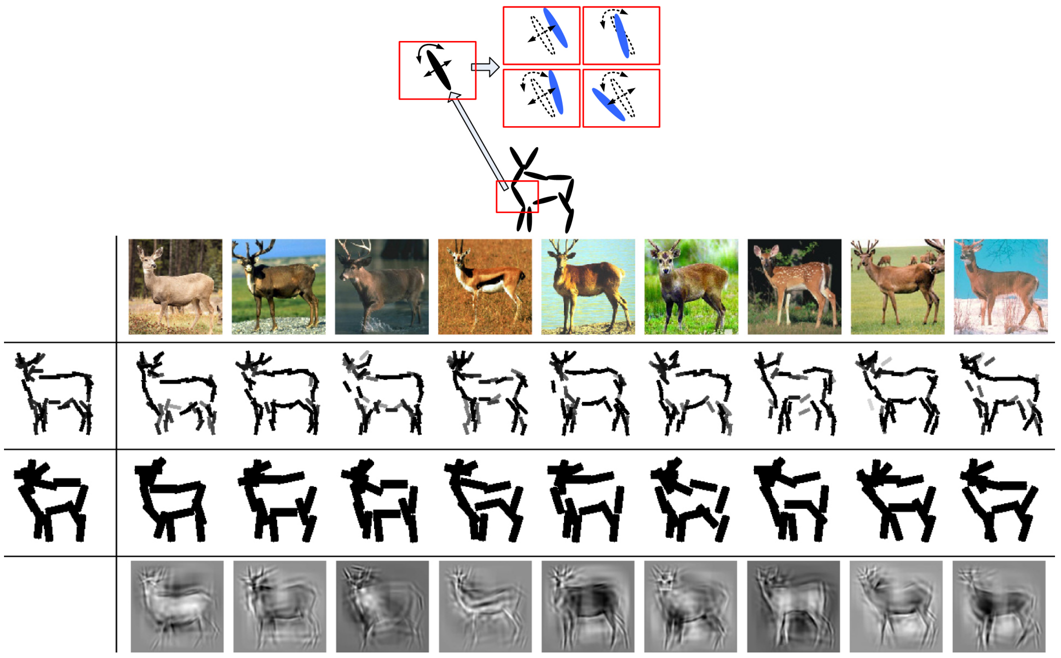


Figure 4.13: (a) An active basis model is a composition of a small number of basis functions, such as Gabor wavelets at selected locations and orientations. Each basis function can perturb its location and orientation within limited ranges. (b) Supervised learning of active basis model from aligned images. 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.

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,α_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 4.13 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 ℓ_2 norm):

$$\sum_{m=1}^{M} |\mathbf{I}_m - \sum_{i=1}^{n} c_{m,i} B_{x_i+\Delta x_{m,i},s,\alpha_i+\Delta\alpha_{m,i}}|^2. \tag{4.11}$$

The prototype algorithm is a greedy algorithm which minimizes the reconstruction error:

0. Initialize $i \leftarrow 0$. For $m = 1, ..., M$, 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 $m = 1, ..., M$, given (x_i, α_i), infer the perturbations in location and orientation by retrieving the arg-max in the local maximum pooling of step 1:

$$(\Delta x_{m,i}, \Delta\alpha_{m,i}) = \arg\max_{\Delta x,\Delta\alpha} |\langle U_m, B_{x_i+\Delta x,s,\alpha_i+\Delta\alpha}\rangle|^2. \tag{4.12}$$

   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:

$$U_m \leftarrow U_m - c_{m,i} B_{x_i+\Delta x_{m,i},s,\alpha_i+\Delta\alpha_{m,i}}. \tag{4.13}$$

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 $\mathbf{B}_m = (B_{x_i+\Delta x_{m,i},s,\alpha_i+\Delta\alpha_{m,i}}, i = 1, \cdots, n)$ 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 $\mathbf{B}_m = (B_{x_i+\Delta x_{m,i},s,\alpha_i+\Delta\alpha_{m,i}}, i = 1, ..., n)$, i.e., $p(\mathbf{I}_m \mid \mathbf{B}_m)$, is obtained by modifying the distribution of natural images $q(\mathbf{I}_m)$ in such a way that we only change the distribution of $C_m = (c_{m,i} = \langle \mathbf{I}_m, B_{x_i+\Delta x_{m,i},s,\alpha_i+\Delta\alpha_{m,i}}\rangle, i = 1, ..., n)$ 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 $p(\mathbf{I}_m \mid \mathbf{B}_m)$ 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, $p(\mathbf{I}_m \mid \mathbf{B}_m) = q(\mathbf{I}_m)p(C_m)/q(C_m)$.

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 $p(\mathbf{I}_m \mid \mathbf{B}_m)/q(\mathbf{I}_m) = p(C_m, U_m)/q(C_m, U_m) = p(C_m)/q(C_m)$. 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 $(c_{m,i} = \langle \mathbf{I}_m, B_{x_i+\Delta x_{m,i},s,\alpha_i+\Delta\alpha_{m,i}}\rangle, i = 1, ..., n)$ are independent given $\mathbf{B}_m$, under both p and q, so

$$p(\mathbf{I}_m \mid \mathbf{B}_m) = q(\mathbf{I}_m)\prod_{i=1}^{n}\frac{p_i(c_{m,i})}{q(c_{m,i})},$$

where $q(c)$ is assumed to be the same for $i = 1, ..., n$ 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 $q(c)$:

$$p(c; \lambda) = \frac{1}{Z(\lambda)}\exp\{\lambda h(|c|^2)\}q(c), \tag{4.14}$$

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 $h(r) \to \xi$ as $r \to \infty$ (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),

$$Z(\lambda) = \int \exp\{\lambda h(r)\}q(c)dc = \mathrm{E}_q[\exp\{\lambda h(r)\}]$$

is the normalizing constant.

$$\mu(\lambda) = \mathrm{E}_\lambda[h(r)] = \int h(r)p(c;\lambda)dc$$

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 $\mathbf{B}_m = (B_{x_i+\Delta x_{m,i},s,\alpha_i+\Delta\alpha_{m,i}}, i = 1, ..., n)$, consider the coefficients obtained by projection: $(c_{m,i}(\mathbf{I}_m) = \langle \mathbf{I}_m, B_{x_i+\Delta x_{m,i},s,\alpha_i+\Delta\alpha_{m,i}}\rangle, i = 1, ..., n)$. Suppose we want to find a probability distribution $p(\mathbf{I}_m \mid \mathbf{B}_m)$ so that $\mathrm{E}[h(|c_{m,i}(\mathbf{I}_m)|^2)] = \mu_i$ for some fixed μ_i, $i = 1, ..., n$, where μ_i can be estimated from the training images. Then among all the distributions that satisfy the constraints on $\mathrm{E}[h(|c_{m,i}(\mathbf{I}_m)|^2)]$, the distribution that is closest to $q(\mathbf{I}_m)$ in terms of the Kullback-Leibler divergence is given by

$$p(\mathbf{I}_m \mid \mathbf{B}_m) = \frac{1}{Z(\Lambda)}\exp\{\sum_{i=1}^n \lambda_i h(|c_{m,i}(\mathbf{I}_m)|^2)\}q(\mathbf{I}_m),$$

where $\Lambda = (\lambda_i, i = 1, ..., n)$, $Z(\Lambda) = \mathrm{E}_q[\exp\{\sum_{i=1}^n \lambda_i h(|c_{m,i}(\mathbf{I}_m)|^2)]$ is the normalizing constant, and Λ is chosen to satisfy the constraints on $\mathrm{E}[h(|c_{m,i}(\mathbf{I}_m)|^2)]$. If we further assume that $c_{m,i}(\mathbf{I}_m)$ are independent of each other for $i = 1, ..., n$ under $q(\mathbf{I}_m)$, then $c_{m,i}(\mathbf{I}_m)$ are also independent under $p(\mathbf{I}_m \mid \mathbf{B}_m)$, and their distributions are of the form (4.14).

In order to choose the nominal template $\mathbf{B}$ and the deformed templates $\{\mathbf{B}_m, m = 1, ..., M\}$, we want $p(\mathbf{I}_m \mid \mathbf{B}_m)$ 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 $\{h(|c_{m,i}|^2), i = 1, ..., n\}$ 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

$$l(\{\mathbf{I}_m\} \mid \mathbf{B}, \{\mathbf{B}_m\}, \Lambda) = \sum_{m=1}^M \log\frac{p(\mathbf{I}_m|\mathbf{B}_m)}{q(\mathbf{I}_m)} \tag{4.15}$$

$$= \sum_{m=1}^M\sum_{i=1}^n \left[\lambda_i h(|\langle \mathbf{I}_m, B_{x_i+\Delta x_{m,i},s,\alpha_i+\Delta\alpha_{m,i}}\rangle|^2) - \log Z(\lambda_i)\right]. \tag{4.16}$$

The expectation of the above log-likelihood ratio is the Kullback-Leibler divergence between $p(\mathbf{I}_m \mid \mathbf{B}_m)$ and $q(\mathbf{I}_m)$.

Given the training images $\{\mathbf{I}_m, m = 1, ..., M\}$, $\sum_{m=1}^M \log q(\mathbf{I}_m)$ is a constant. Thus maximizing the log-likelihood ratio $\sum_{m=1}^M \log p(\mathbf{I}_m \mid \mathbf{B}_m, \Lambda)/q(\mathbf{I}_m)$ is equivalent to maximizing the log-likelihood $\sum_{i=1}^M \log p(\mathbf{I}_m \mid \mathbf{B}_m, \Lambda)$. 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 $m = 1, ..., M$, initialize the response maps $R_m(x, \alpha) \leftarrow \langle \mathbf{I}_m, B_{x,s,\alpha} \rangle$ for all (x, α).

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 $m = 1, ..., M$, given (x_i, α_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 λ_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, α_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 $\mu(\lambda_i) = \text{E}_{\lambda_i}[h(|c|^2)] = \int h(|c|^2)p(c; \lambda_i)dc$ 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. λ_i is solved so that $\mu(\lambda_i)$ matches the empirical average of $h(|c_{m,i}|^2)$, $m = 1, ..., M$. The function $\mu()$ can be computed and stored over a discrete set of equal-spaced values, so that λ_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 λ_i. The estimated λ_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, α_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, α).

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:

$$l(X) = \sum_{i=1}^{n} \left[\lambda_i \max_{\Delta x, \Delta\alpha} h(|\langle \mathbf{I}, B_{X+x_i+\Delta x, s, \alpha_i+\Delta\alpha}\rangle|^2) - \log Z(\lambda_i)\right]. \tag{4.17}$$

2. Find maximum likelihood location $\hat{X} = \arg\max_X l(X)$. For $i = 1, ..., n$, 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 $(B_{\hat{X}+x_i+\Delta x_i, s, \alpha_i+\Delta\alpha_i}, i = 1, ..., n)$, 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.

4.4 Compositional Sparse Coding

4.4.1 Sparsity and Composition

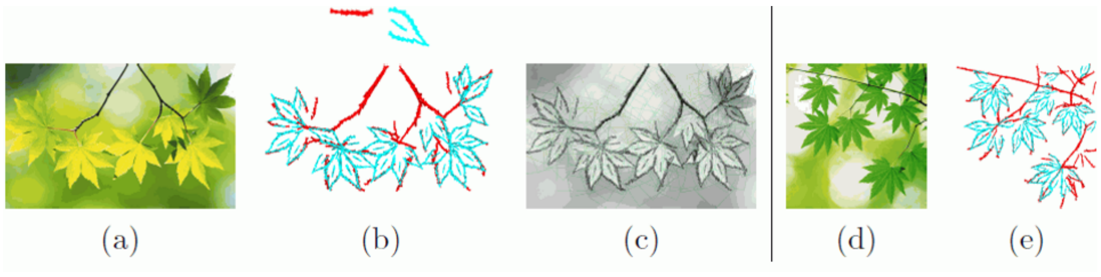


Figure 4.14: Unsupervised learning of compositional sparse code (a,b,c) and using it for recognition and segmentation (d,e). (a) Training image of 480 × 768 pixels. (b) Above: 2 compositional patterns (twig and leaf) in the form of shape templates learned from the training image. Each constituent Gabor 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 × 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.

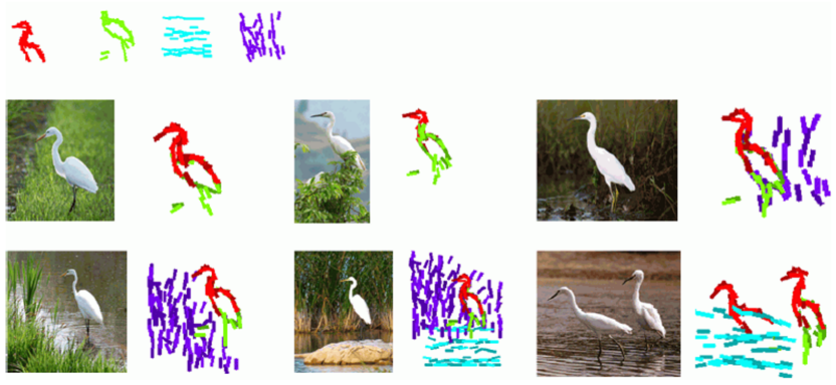


Figure 4.15: Four compositional patterns (templates) are learned from 20 training images (only 6 of them are shown in this figure). The training images are not registered or otherwise annotated. The size of each template is 100 $\times$ 100 pixels. The number of basis functions in each template is no more than 40 and is automatically determined.

The goal of this section is to develop a compositional sparse code for natural images. Figure 4.14 illustrates the basic idea. We start with a dictionary of Gabor wavelets centered at a dense collection of locations and tuned to a collection of scales and orientations. In Figure 4.14, each Gabor wavelet is illustrated by a bar at the same location and with the same length and orientation as the corresponding wavelet. Figure 4.14.(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 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 4.15 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 ℓ_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.

4.4.2 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.

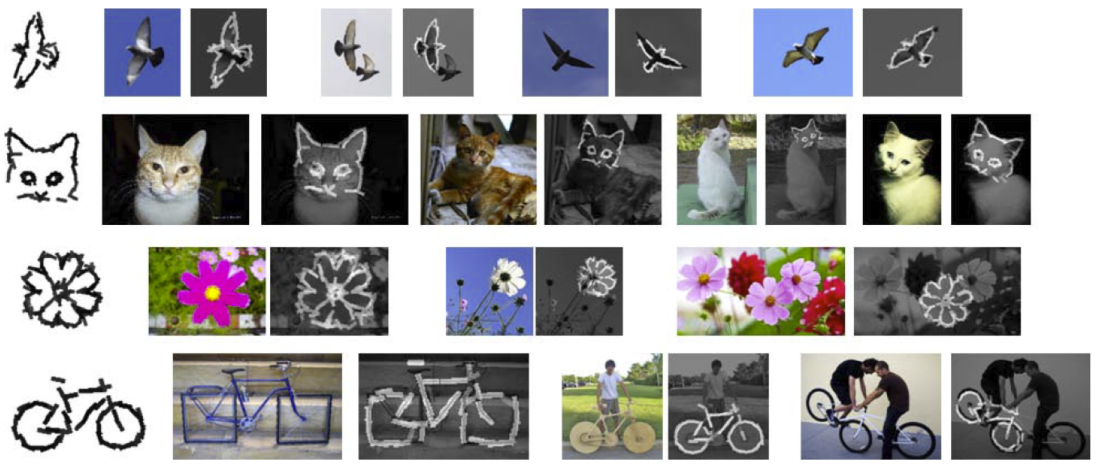


Figure 4.16: Objects appear at different locations, scales and orientations in the training images. In each row, the first plot displays the nominal active basis template. The rest of the row displays some examples of training images and the suppositions of the spatially translated, scaled, rotated and 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:

$$\begin{aligned} \mathbf{I}_m &= \sum_{i=1}^n c_{m,i} B_{X_m + x_i + \Delta x_{m,i}, s, \alpha_i + \Delta\alpha_{m,i}} + U_m \\ &= C_m \mathbf{B}_{X_m} + U_m, \end{aligned} \tag{4.18}$$

where $\mathbf{B}_{X_m} = (B_{X_m + x_i + \Delta x_{m,i}, s, \alpha_i + \Delta\alpha_{m,i}}, i = 1, ..., n)$ is the deformed template spatially translated to X_m, $C_m = (c_{m,i}, i = 1, ..., n)$, and $C_m B_{X_m} = \sum_{i=1}^n c_{m,i} B_{X_m + x_i + \Delta x_{m,i}, s, \alpha_i + \Delta\alpha_{m,i}}$ 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):

$$\begin{aligned} l(\mathbf{I}_m \mid \mathbf{B}_{X_m}) &= \log \frac{p(\mathbf{I}_m \mid \mathbf{B}_{X_m})}{q(\mathbf{I}_m)} \\ &= \sum_{i=1}^{n} \left[\lambda_i \max_{\Delta x, \Delta\alpha} h(|\langle \mathbf{I}_m, B_{X_m + x_i + \Delta x, s, \alpha_i + \Delta\alpha} \rangle|^2) - \log Z(\lambda_i) \right]. \end{aligned} \tag{4.19}$$

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 (4.18) is

$$\mathbf{I}_m = C_m \mathbf{B}_{X_m, S_m, A_m} + U_m, \tag{4.20}$$

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 (4.19). Figure 4.16 illustrates the basic idea of representation (4.20). 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, $\{\mathbf{B}^{(t)}, t = 1, ..., T\}$, 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:

$$\mathbf{I}_m = \sum_{k=1}^{K_m} C_{m,k} \mathbf{B}^{(t_{m,k})}_{X_{m,k}, S_{m,k}, A_{m,k}} + U_m, \tag{4.21}$$

where each $\mathbf{B}^{(t_{m,k})}_{X_{m,k}, S_{m,k}, A_{m,k}}$ 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 (4.21), 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

$$\sum_{m=1}^{M} \sum_{k=1}^{K_m} \left[l(\mathbf{I}_m \mid \mathbf{B}^{(t_{m,k})}_{X_{m,k}, S_{m,k}, A_{m,k}}) \right]. \tag{4.22}$$

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 $\mathbf{B}^{(t_{m,k})}_{X_{m,k}, S_{m,k}, A_{m,k}}$ 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 (4.21) is $K_m g$. In fact, we can unpack model (4.21) 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 $\mathbf{B}^{(t_{m,k})}_{X_{m,k}, S_{m,k}, A_{m,k}}$, 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 $\mathbf{B}^{(t_{m,k})}_{X_{m,k},S_{m,k},A_{m,k}}$ 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 ρD 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 $\{\mathbf{B}^{(t)}, t = 1, ..., T\}$. 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 penalized log-likelihood ratio (**??**).

[I.0] Initialize the maps of template matching scores for all (X, S, A, t):

$$\mathbf{R}^{(t)}_m(X, S, A) \leftarrow l(\mathbf{I}_m \mid \mathbf{B}^{(t)}_{X,S,A}) - n^{(t)}\gamma,$$

where $n^{(t)}$ is the number of basis functions in the t-th template in the dictionary and γ is a constant controling 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:

$$(X_{m,k}, S_{m,k}, A_{m,k}, t_{m,k}) = \arg\max_{X,S,A,t} \mathbf{R}^{(t)}_m(X, S, A).$$

[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 $\mathbf{B}^{(t_{m,k})}_{X_{m,k},S_{m,k},A_{m,k}}$, then for all (X, S, A, t), if X is within a distance ρD from $X_{m,k}$, then set the response $\mathbf{R}^{(t)}_m(X, S, A) \leftarrow -\infty$ (default setting: $\rho = .4$).

[I.3] Stop if all $\mathbf{R}^{(t)}_m(X, S, A, t) < 0$. 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 $U_m \leftarrow U_m - C_m\mathbf{B}^{(t_{m,k})}_{X_{m,k},S_{m,k},A_{m,k}}$ 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 $t = 1, ..., T$, 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 $(t_{m,k}, X_{m,k}, S_{m,k}, A_{m,k}, k = 1, ..., K_m)$.

[II.0] Image patch cropping. For each $\mathbf{I}_m$, go through all the selected templates $\{\mathbf{B}^{(t_{m,k})}_{X_{m,k},S_{m,k},A_{m,k}}, \forall k\}$ 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 $\mathbf{B}^{(t_{m,k})}_{X_{m,k},S_{m,k},A_{m,k}}$.

[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.

4.5 The Sparse FRAME Model

The FRAME model is a Markov random field model [13,60]. 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 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 [205, 207] 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).

4.5.1 Dense FRAME

We start from the non-stationary or spatially inhomogeneous FRAME model [200, 205, 207] 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 [15,42,144]). The model is a random field of the following form:

$$p(\mathbf{I}; w) = \frac{1}{Z(w)} \exp\left[\sum_{k=1}^{K} \sum_{x \in \mathcal{D}} w_{k,x} h(\langle \mathbf{I}, B_{k,x} \rangle)\right] q(\mathbf{I}). \tag{4.23}$$

The above model is a simple generalization of FRAME model (10.3) , 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. $h()$ is a pre-specified rectification function. In [205] , $h(r) = |r|$, 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

$$q(\mathbf{I}) = \frac{1}{(2\pi\sigma^2)^{D/2}} \exp\left[-\frac{1}{2\sigma^2} ||\mathbf{I}||^2\right], \tag{4.24}$$

where again D counts the number of pixels in the image domain $\mathcal{D}$.

4.5.2 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 $\{B_k, k = 1, ..., K\}$, and where $B_{k,x}$ is a spatially translated copy of B_k to position x. We can represent an image $\mathbf{I}$ by

$$\mathbf{I} = \sum_{k,x} c_{k,x} B_{k,x} + \epsilon, \tag{4.25}$$

where $c_{k,x}$ are the coefficients, and ϵ 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 [43, 144].

The sparsification of $c_{k,x}$, i.e., the selection of the basis functions, can be accomplished by matching pursuit [126] or basis pursuit/Lasso [19, 181]. Using a Lasso-like objective function, the dictionary of basis functions $\{B_k\}$ can be learned from a collection of training images [42, 144]. It is sometimes called sparse component analysis [39]. It can be considered a generalization of factor analysis. For natural images, the basis functions learned resemble the Gabor and DoG wavelets in Figure 11.8.

4.5.3 Maximum likelihood learning

The basic learning algorithm estimates the parameters $w = (w_{k,x}, \forall k, x)$ from a set of aligned training images $\{\mathbf{I}_i, i = 1, ..., n\}$ 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

$$L(w) = \frac{1}{n}\sum_{i=1}^{n} \log p(\mathbf{I}_i; w), \tag{4.26}$$

whose partial derivatives are

$$\frac{\partial L(w)}{\partial w_{k,x}} = \frac{1}{n}\sum_{i=1}^{n} h(\langle \mathbf{I}_i, B_{k,x}\rangle) - \mathrm{E}_w\left[h(\langle \mathbf{I}, B_{k,x}\rangle)\right], \tag{4.27}$$

where E_w denotes expectation with respect to $p(\mathbf{I}; w)$ in (4.23). This expectation can be approximated by Monte Carlo integration. Thus, w can be computed by the stochastic gradient ascent algorithm [155, 214]

$$w_{k,x}^{(t+1)} = w_{k,x}^{(t)} + \gamma_t \left[\frac{1}{n}\sum_{i=1}^{n} h(\langle \mathbf{I}_i, B_{k,x}\rangle) - \frac{1}{\tilde{n}}\sum_{i=1}^{\tilde{n}} h(\langle \tilde{\mathbf{I}}_i, B_{k,x}\rangle)\right], \tag{4.28}$$

where γ_t is the step size or the learning rate, and $\{\tilde{\mathbf{I}}_i, i = 1, ..., \tilde{n}\}$ are the synthetic images sampled from $p(\mathbf{I}; w^{(t)})$ using MCMC, such as Hamiltonian Monte Carlo [138] or the Gibbs sampler [59]. $\tilde{n}$ is the total number of independent parallel Markov chains that sample from $p(\mathbf{I}; w^{(t)})$.

4.5.4 Generative boosting

Model (4.23) 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 [207] of the epsilon-boosting algorithm [51, 54] (see also [31, 48, 197, 201]). The algorithm starts from $w = 0$, the zero vector. At the t-th iteration, let

$$\Delta_{k,x} = \frac{1}{n}\sum_{i=1}^{n} h(\langle \mathbf{I}_i, B_{k,x}\rangle) - \frac{1}{\tilde{n}}\sum_{i=1}^{\tilde{n}} h(\langle \tilde{\mathbf{I}}_i, B_{k,x}\rangle) \tag{4.29}$$

be the Monte Carlo estimate of $\partial L(w)/\partial w_{k,x}$, where again $\{\tilde{\mathbf{I}}_i, i = 1, ..., \tilde{n}\}$ are the synthetic images sampled from the current model. We select

$$(\hat{k}, \hat{x}) = \arg\max_{k,x} |\Delta_{k,x}|, \tag{4.30}$$

and update $w_{\hat{k},\hat{x}}$ by

$$w_{\hat{k},\hat{x}} \leftarrow w_{\hat{k},\hat{x}} + \gamma_t \Delta_{\hat{k},\hat{x}}, \tag{4.31}$$

where γ_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 4.17 for an illustration.

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

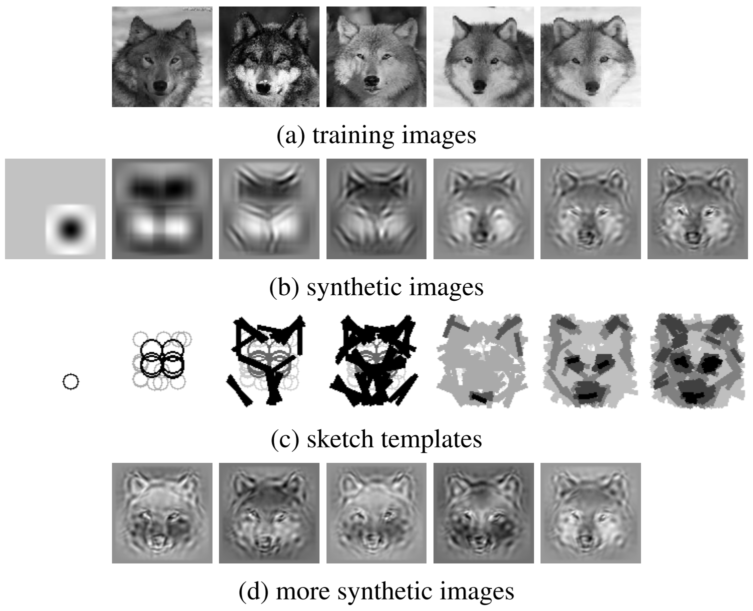


Figure 4.17: Learning process of the generative boosting. (a) observed training images (100×100 pixels) from which the random field model is learned. (b) a sequence of synthetic images generated by the learned model as more and more wavelets are induced into the model. The numbers of the selected wavelets 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.

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 [54, 75] has an interesting relationship with the ℓ_1 regularization in the Lasso [181] and basis pursuit [19]. As pointed out by [156], under a monotonicity condition (e.g., the components of w keep increasing), such an algorithm approximately traces the solution path of the ℓ_1 regularized minimization of

$$-L(w) + \rho \|w\|_{\ell_1}, \tag{4.32}$$

where the regularization parameter ρ 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.

4.5.5 Sparse model

After selecting m wavelets, we have the following sparse FRAME model:

$$p(\mathbf{I}; \mathbf{B}, w) = \frac{1}{Z(w)} \exp\left[\sum_{j=1}^{m} w_j h(\langle \mathbf{I}, B_{k_j, x_j} \rangle)\right] q(\mathbf{I}), \tag{4.33}$$

where $\mathbf{B} = (B_j = B_{k_j,x_j}, j = 1, ..., m)$ is the set of wavelets selected from the dictionary, and $w_j = w_{k_j,x_j}$.

In model (4.33), m is much smaller than D, the number of pixels. Thus, we can represent $\mathbf{I}$ by

$$\mathbf{I} = \sum_{j=1}^{m} c_j B_{k_j,x_j} + \epsilon, \tag{4.34}$$

where $C = (c_j, j = 1, ..., m)^\top$ are the least square regression coefficients of $\mathbf{I}$ on $\mathbf{B} = (B_j, j = 1, ..., m)$, i.e., $C = (\mathbf{B}^\top\mathbf{B})^{-1}\mathbf{B}^\top\mathbf{I}$, and ϵ is the residual image. The distribution of C under $p(\mathbf{I}; \mathbf{B}, w)$ is

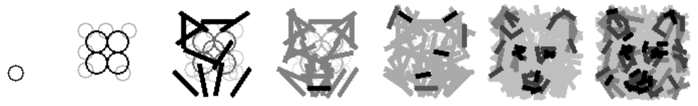
$$p_C(C; w) = \frac{1}{Z(w)} \exp\left[\langle w, h(\mathbf{B}^\top\mathbf{B}C)\rangle\right] q_C(C), \tag{4.35}$$

where $q_C(C)$ is the distribution of C under $q(\mathbf{I})$, and the transformation $h()$ is applied element-wise. Thus, $p(\mathbf{I}; B, w)$ in (4.33) can be written as a wavelet sparse coding model (4.34) and (4.35). The forms of (4.33) and (4.34) show that the selected wavelets $\{B_j\}$ serve both as filters and basis functions. The sparse coding form of the model (4.34) and (4.35) 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 [59], and then generating $\tilde{\mathbf{I}}_i$ according to (4.34).

Model (4.34) suggests that we can also select the wavelets by minimizing

$$\sum_{i=1}^{n} \|\mathbf{I}_i - \sum_{j=1}^{m} c_{i,j} B_{k_j,x_j}\|^2, \tag{4.36}$$

using a shared matching pursuit method [205]. See Figure 4.18 for an illustration. We can also allow the selected wavelets to perturb their locations and orientations to account for deformations [200].


(a) sketch templates

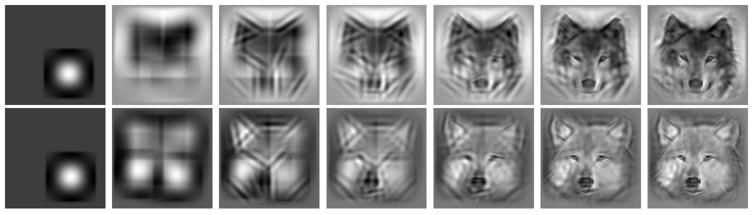

(b) reconstructed images

Figure 4.18: 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 × 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 4.17(a).

The sparse FRAME model can be used for unsupervised learning tasks such as model-based clustering [47]. 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 [83, 205].

The sparse FRAME model merges two important research themes in image representation and modeling, namely, Markov random fields [13, 60] and wavelet sparse coding [42, 144].

The wavelets can be mapped to the first layer filters of a ConvNet [111] 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.

5

Primal Sketch: Integrating Textures and Textons

In his monumental book [127], Marr inherited Julesz's texton [96] notion and proposed the concept of image primitives as basic perceptual tokens, such as edges, bars, junctions, and terminators. Inspired by the Nyquest 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.

5.1 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 near distance whose positions and shapes can be clearly perceived. In contrast, the textural part is composed of objects at far distance whose structures becomes 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.

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.

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 by 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 is shown to follow some MRF models called FRAME.

5.2 The Two Layer Model

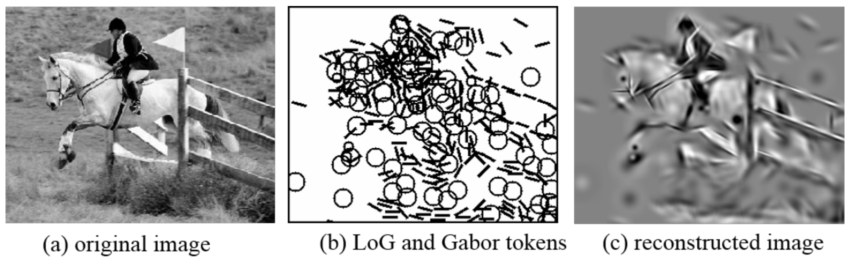


Figure 5.1: A sparse coding example computed by matching pursuit. (b) is a symbolic representation where each base B_k is represented by a bar at the same location, with the same elongation and orientation. The isotropic LOG bases are represented by a circle.

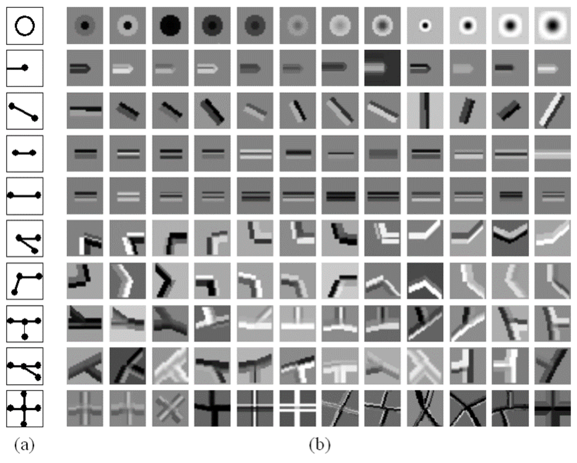


Figure 5.2: A collection of local structure elements employed by the model. There are eight types of elements: blobs, end points, edges, ridges, multi-ridges, corners, junctions and crosses. (a) The symbolic representation. (b) The photometric representation.

According to the model, the image is generated as a mosaic as follows: the image lattice Λ is divided into a structure domain $\Lambda_{\mathrm{s}k}$ and a textured domain $\Lambda_{\mathrm{n}sk}$. The image intensities on the structure domain are

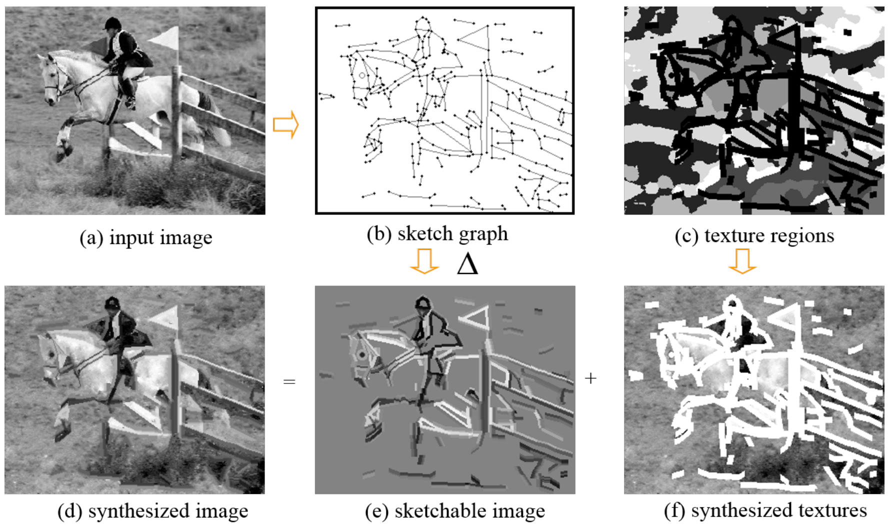


Figure 5.3: Primal sketch model. (a) Observed image. (b) "Sketchable" part is described by a geometric sketch graph. (c) The sketchable part of the image. (d) Fill in the "non-sketchable" part by matching feature statistics.

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.

5.2.1 Structure domain

The model for the structure domain of the image is

$$\mathbf{I}(x,y) = \sum_{i=1}^{n} B(x,y|\theta_i) + \epsilon(x,y), \quad (x,y) \in D_{\text{sk}}, \quad i = 1, ..., n. \tag{5.1}$$

The coding functions $B(x,y|\theta_i)$ are used to represent edge and ridge segments (as well as blobs) in the image, where θ_i are geometric and photometric parameters of these coding functions. Let $D_{sk,i}$ be the set of pixels coded by $B(x,y \mid \theta_i)$. They do not overlap each other except over the small number of pixels where they join each other to form corners and junctions. Therefore, $B(x,y|\theta_i)$ is similar to coding vectors in vector quantization.

An edge segment is modeled by a 2D function that is constant along the edge, and has a profile across the edge. Specifically,

$$B(x,y \mid \theta) = f(-(x-u)\sin\alpha + (y-v)\cos\alpha), \tag{5.2}$$

where

$$-l < (x-u)\cos\alpha + (y-v)\sin\alpha \leq l,$$
$$-w \leq -(x-u)\sin\alpha + (y-v)\cos\alpha \leq w.$$

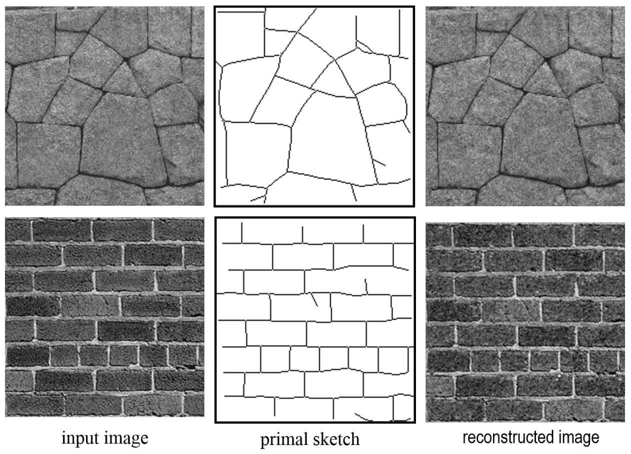


Figure 5.4: Examples of primal sketch model. (a) Observed image. (b) Sketch graph. (c) Synthesized image from the fitted model.

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 α.

For the profile function $f()$, let $f_0(x) = -1/2$ for $x < 0$ and $f_0(x) = 1/2$ for $x \geq 0$, and let $g_s()$ be a Gaussian function of standard deviation s. Then $f() = a + bf_0() * g_s()$. 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 Zucke, the parameter s can be determined by the distance between the two extrema of the second derivative $f''()$.

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. θ captures geometric and photometric aspects of an edge explicitly, and the coding function is non-linear in θ.

A ridge segment has the same functional form, where the profile $f()$ 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, $B(x, y \mid \theta) = f(\sqrt{(x-u)^2 + (y-v)^2} - r)$, where $(x-u)^2 + (y-v)^2 \leq R^2$, and again $f() = a + bf_0() * g_s()$ 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 end point of a ridge is modeled by a half blob.

See Figure (5.2) for a sample of local structure elements, which are the coding functions and their combinations. There are eight types of elements: blobs, end points, edges, ridges, multi-ridges, corners, junctions and crosses. Figure (5.2.a) shows the symbolic representations of these elements. Figure (5.2.b) displays the image patches of these elements.

Let $S_{\text{sk}} = (\theta_i, i = 1, ..., n)$ 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 end point has degree 1, a corner has degree 2, a T-junction has degree 3, and a cross has degree 4. We do not allow nodes with more than 4 arms. S_{sk} is regularized by a simple spatial prior model:

$$p(S_{\mathrm{sk}}) \propto \exp\{-\sum_{d=0}^{4} \lambda_d |V_d|\}, \tag{5.3}$$

where $|V_d|$ is the number of nodes with d arms. The prior probability or the energy term $\gamma_{\mathrm{sk}}(S_{\mathrm{sk}}) = \sum_{d=0}^{4} \lambda_d |V_d|$ penalizes free end points by setting λ_{sk} at a large value.

5.2.2 Texture domain

The texture domain D_{nsk} is segmented into m regions of homogenous texture patterns, $D_{\mathrm{nsk}} = \cup_{j=1}^{m} D_{\mathrm{nsk},j}$. Within each region j, we pool the marginal histograms of the responses from the K filters, $h_j = (h_{j,k}, k = 1, ..., K)$, where

$$h_{j,k,z} = \frac{1}{|D_{\mathrm{nsk},j}|} \sum_{(x,y) \in D_{\mathrm{nsk},j}} \delta(z; F_k * \mathbf{I}(x,y)), \tag{5.4}$$

where z indexes the histogram bins, and $\delta(z;x) = 1$ if x belongs to bin z, and $\delta(z;x) = 0$ otherwise.

According to the previous section, this is equivalent to a Markov random field model for each texture region:

$$p(\mathbf{I}_{D_{\mathrm{nsk},j}}) \propto \exp\{-\sum_{(x,y) \in D_{\mathrm{nsk},j}} \sum_{k=1}^{K} \phi_{j,k}(F_k * \mathbf{I}(x,y))\}. \tag{5.5}$$

These Markov random fields have the structure domain as boundary conditions, because when we apply filters F_k on the pixels in D_{nsk}, these filters may also cover some pixels in D_{sk}. These Markov random fields in-paint the texture domain D_{nsk} while interpolating the structure domain D_{sk}, and the in-painting is guided by the marginal histograms of linear filters within each region.

Let $S_{\mathrm{nsk}} = (D_{\mathrm{nsk},j}, j = 1, ..., m)$ denotes the segmentation of the texture domain. S_{nsk} follows a prior model $p(S_{\mathrm{nsk}}) \propto \exp\{-\gamma_{\mathrm{nsk}}(S_{\mathrm{nsk}})\}$, for instance, $\gamma_{\mathrm{nsk}}(S_{\mathrm{nsk}}) = \rho m$ to penalize the number of regions.

5.2.3 Integrated model

Formally, we can integrate the structure model (5.1) and the texture model (5.5) into a probability distribution. Our inspiration for such an integration comes from the model of Mumford and Shah [**?**]. In their method, the prior model for the noiseless image can be written as

$$p(\mathbf{I}, S) = \frac{1}{Z} \exp\{-\sum_{(x,y) \in D/S} \lambda |\nabla \mathbf{I}(x,y)|^2 - \gamma |S|\}, \tag{5.6}$$

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 (5.6), 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 $S = (S_{\mathrm{sk}}, S_{\mathrm{n}sk})$, we have

$$\begin{aligned} p(\mathbf{I}, S) = \frac{1}{Z}\exp\{ \quad & - \sum_{i=1}^{n} \sum_{(x,y)\in D_{\mathrm{sk},i}} \frac{1}{2\sigma^2}(\mathbf{I}(x,y) - B_i(x,y \mid \theta_i))^2 - \gamma_{\mathrm{s}k}(S_{\mathrm{sk}}) \\ & - \sum_{j=1}^{m} \sum_{(x,y)\in D_{\mathrm{n}sk,j}} \sum_{k=1}^{K} \phi_{j,k}(F_k * \mathbf{I}(x,y)) - \gamma_{\mathrm{n}sk}(S_{\mathrm{n}sk})\}. \end{aligned} \tag{5.7}$$

Compared to Mumford-Shah model, model (5.7) is more sophisticated in both the structure part and texture part.

The sketch pursuit algorithm is a greedy algorithm. It consists of the following phases:

1. *Phase*: an edge and ridge detector based on linear filters is run to give an initialization for the sketch graph.
2. *Phase*: a greedy algorithm is used to determine the sketch graph but without using the spatial prior model.
3. *Phase*: 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*: 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.

5.3 Hybrid Image Templates

The active basis templates target the sketch parts of the images. In this section, we shall further add the texture appearances of the images, so the templates become hybrid image templates HIT.

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 Fig. 5.5. We both to describe an image.

For instance, the image in Fig. 5.6 consists 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. 5.7 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.

The modeling and learning strategy of hybrid template is similar to active basis. For each type of features, 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 common shape outline, but do not have common texture or color, the hedgehog in Fig. 5.6 has distinct

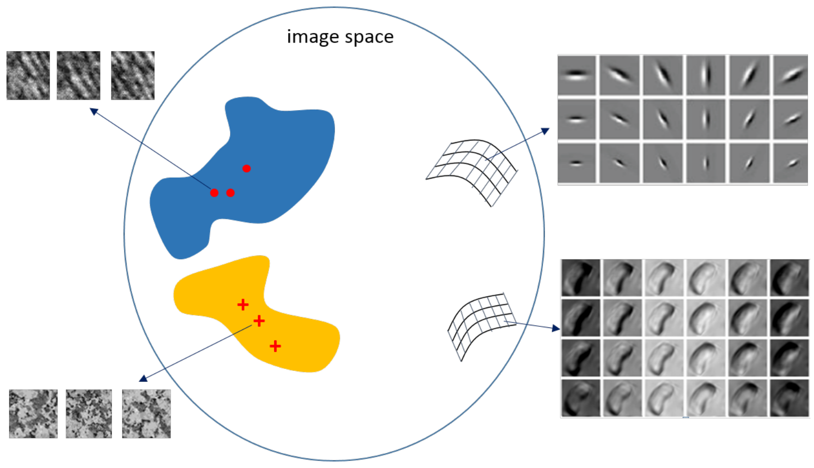


Figure 5.5: In the space of image patches, there are simple shape primitives, and there are also stochastic texture patterns.

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.

5.3.1 Representation

Let Λ be the image lattice for the object template which is typically of 150×150 pixels. This template will undergo a similarity transform to align with object instance in images. The lattice is decomposed into a set of K patches $\{\Lambda_k, k = 1, 2..., K\}$ 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 Λ for two reasons:

- Certain pixels on Λ 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,

$$\text{HIT} = (\,\{\Lambda_k, \ell_k, \{B_k \text{ or } \mathbf{h}_k\}, \delta_k :, k = 1, 2, ..., K\},\, \Theta) \tag{5.8}$$

where

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.

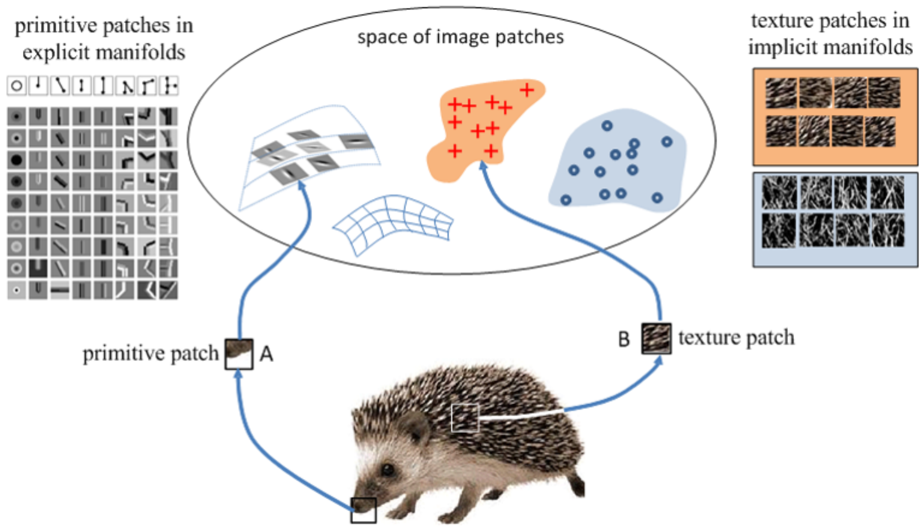

primitive patches in explicit manifolds
space of image patches
texture patches in implicit manifolds
primitive patch
A
B
texture patch

Figure 5.6: An image of an object consists of patches of simple primitives and patches of textures.

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. $\Theta = \{\lambda_k, z_k : k = 1, 2, ..., K\}$ are the parameters of the probabilistic model p (to be discussed in the subsection). λ_k, z_k are the linear coefficient and normalizing constant for the k-th patch.

5.3.2 Prototypes, ϵ - 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),

$$\Omega(B_k) = \{\mathbf{I}_{\Lambda_k} : \mathbf{I}_{\Lambda_k} = c_k B_k + \epsilon\}. \tag{5.9}$$

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),

$$\Omega(\mathbf{h}_k) = \{\mathbf{I}_{\Lambda_k} : H(\mathbf{I}_{\Lambda_k}) = \mathbf{h}_k + \epsilon\}. \tag{5.10}$$

$H(\mathbf{I}_{\Lambda_k})$ 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,

$$\rho^{\text{ex}}(\mathbf{I}_{\Lambda_k}) = \|\mathbf{I}_{\Lambda_k} - cB_k\|^2 \tag{5.11}$$

while in $\Omega(h_k)$, the distance is measured in the projected histogram space with norm.

$$\rho^{\text{im}}(\mathbf{I}_{\Lambda_k}) = \|H(\mathbf{I}_{\Lambda_k}) - \mathbf{h}_k\|^2 \tag{5.12}$$

Intuitively, we may view $\Omega(B_k)$ and $\Omega(\mathbf{h}_k)$ as ϵ-balls centered at the prototypes B_k and $\mathbf{h}_k$, respectively, with different metrics. Each ϵ-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: $\Omega(HIT) = \prod_{k=1}^{K} \Omega_k$, on which a probability model is concentrated. Due to statistical fluctuations in small patches, these ϵ-balls have

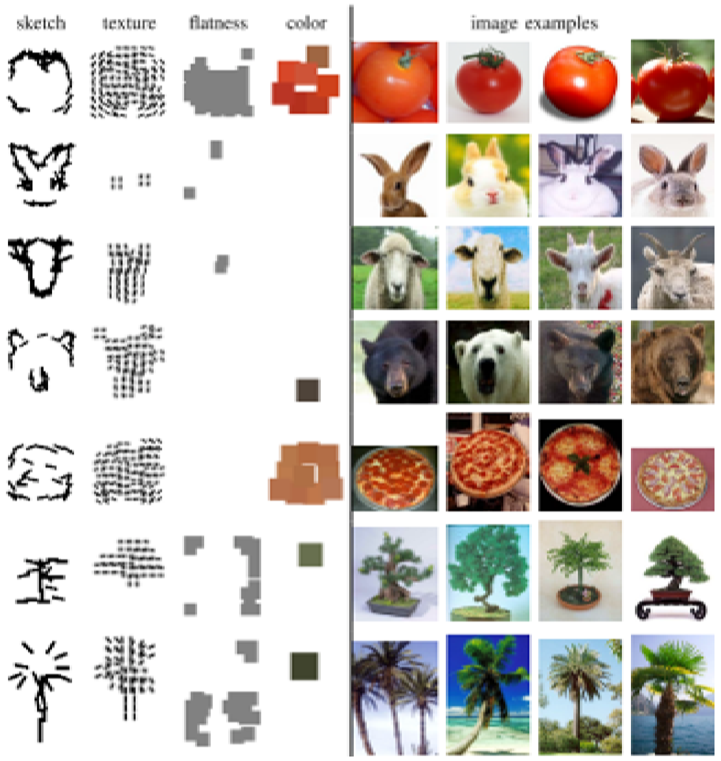


Figure 5.7: The hybrid templates consist of sketch, texture, flatness, and color features.

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 $\Omega(\mathbf{h}_k)$.

$$r(\mathbf{I}_{\Lambda_k}) = \mathrm{S}(\rho(\mathbf{I}_{\Lambda_k})), \tag{5.13}$$

where ρ can be either ρ^{ex} or ρ^{im}. $S(x)$ is a saturation function with maximum at $x = 0$:

$$\mathrm{S}(x) = \tau \left(\frac{2}{1 + e^{-2(\eta - x)/\tau}} - 1 \right), \tag{5.14}$$

with shape parameters τ and η. We set $\tau = 6$ and η 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 $\mathbf{I}_{\Lambda_k}$.

5.3.3 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 in 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 Λ, we first transform it into a HSV-space with HS being the chromatic information and V the gray level image. We apply a common set of filters Δ to the gray level image. The dictionary Δ 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 δ_k,

$$r^{\text{skt}}(\mathbf{I}_{\Lambda_k}) = \max_{\delta x, \delta y, \delta\theta} \mathrm{S}(\|\mathbf{I} - cB_{x+\delta x, y+\delta y, o+\delta o}\|^2). \tag{5.15}$$

The local maximum pooling is proposed by [**?**] 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 **??**. 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

$$r^{\text{txt}}(\mathbf{I}_{\Lambda_k}) = \mathrm{S}(\|H(\mathbf{I}_{\Lambda_k}) - \mathbf{h}\|^2), \tag{5.16}$$

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 strength 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,

$$H_o(\mathbf{I}_{\Lambda_k}) = \frac{1}{|\Lambda_k|} \sum_{(x,y)\in\Lambda_k} a(|F_o * \mathbf{I}_{\Lambda_k}|^2). \tag{5.17}$$

Thus, we obtain the oriented histogram for all filters as a $|\mathcal{O}|$-vector,

$$H(\mathbf{I}_{\Lambda_k}) = (H_1, ..., H_{|\mathcal{O}|}). \tag{5.18}$$

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,

$$H(\mathbf{I}_{\Lambda_k}) = \sum_{F\in\Delta^{\text{flt}}} \sum_{(x,y)\in\Lambda_k} b(|F_o * \mathbf{I}_{\Lambda_k}|^2). \tag{5.19}$$

$b()$ 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 **??** we plot the four functions $a()$, $b()$, $\mathbf{1}()$ and $S()$ for comparison.

Then the flatness response is defined as,

$$r^{\text{flt}}(\mathbf{I}_{\Lambda_k}) = \mathrm{S}(H(\mathbf{I}_{\Lambda_k}) - h). \tag{5.20}$$

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 $\mathbf{h}$,

$$r^{\text{clr}}(\mathbf{I}_{\Lambda_k}) = \mathrm{S}(\|H^{\text{clr}}(\mathbf{I}_{\Lambda_k}) - \mathbf{h}\|^2). \tag{5.21}$$

In summary, a HIT template consists of K prototypes $\{B_k \text{ or } \mathbf{h}_k, k = 1, \cdots, K\}$ for sketch, texture/gradient, flatness, and color patches respectively which define K-subspaces (or ϵ-balls) $\Omega(B_K)$ or $\Omega(\mathbf{h}_k)$ of varying dimensions. These ϵ-balls quantize the image space with different metrics. An input image $\mathbf{I}$ on lattice Λ is then projected to the HIT and is represented by a vector of responses:

$$\mathbf{I} \to (r_1, r_2, ..., r_K)$$

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.

5.3.4 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,

$$q = p_0 \to p_1 \to p_2 \to \cdots \to p_K \approx f. \tag{5.22}$$

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 $KL(f||p)$ monotonically.

The main input to the learning algorithm is a set of positive examples

$$D^+ = \{\mathbf{I}_1, ..., \mathbf{I}_n\} \sim f,$$

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

$$D^- = \{\mathbf{J}_1, ..., \mathbf{J}_N\} \sim \text{ reference distribution } q.$$

The negative examples are only used for pooling marginal histograms of one-dimensional feature responses in a pre-computation step.

The image lattice Λ 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 Δ, 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.

$$\Omega_{\text{cand}} = \{\Lambda_j, \ell_j, \{B_j \text{ or } \mathbf{h}_j\} : j = 1, 2, ..., M\}. \tag{5.23}$$

From Ω_{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 $p = p_{k-1}$:

$$\text{HIT}_{k-1} = (\,\{\Lambda_j, \ell_j, B_j \text{ or } \mathbf{h}_j, \delta_j, j = 1, ..., k-1\},\ \Theta_{k-1}).$$

Consider a new candidate patch Λ_k in Ω_{cand} and its responses on n positive examples and N negative examples:

$$\{r^+_{k,i},\ i = 1, ..., n\} \quad \{r^-_{k,i},\ i = 1, ..., N\}. \tag{5.24}$$

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 Λ_1 and Λ_2 have little overlap; ii) Λ_1 and Λ_2 are from different scales. If at the k-th step we have removed from Ω_{cand} all the candidate patches that overlap with selected patches, then r_k is roughly uncorrelated with all the previously selected responses $r_1, ..., r_{k-1}$. 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 Λ_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.

$$p^*_k = \arg\min \quad KL(p_k | p_{k-1}) \tag{5.25}$$
$$s.t. \qquad E_{p_k}[r_k] = E_f[r_k] \tag{5.26}$$

By solving the Euler-Lagrange equation with Lagrange multipliers $\{\lambda_j\}$ and γ,

$$\frac{\partial}{\partial p_k}\Big\{\sum_{\mathbf{I}} p_k(\mathbf{I}) \log \frac{p_k(\mathbf{I})}{p_{k-1}(\mathbf{I})} + \lambda_k(E_{p_k}[r_j] - E_f[r_j]) \\ + \gamma(\sum_{\mathbf{I}} p_k(\mathbf{I}) - 1)\Big\} = 0.$$

Thus we have,

$$p_k(\mathbf{I}) = p_{k-1}(\mathbf{I}) \frac{1}{z_k} \exp\{-\lambda_k r_k(\mathbf{I})\}. \tag{5.27}$$

$z_k = E_q\left[\exp\{\lambda_k r_k(\mathbf{I}_{\Lambda_k})\}\right]$ is a normalizing constant. This can be estimated by the negative samples,

$$z_k \approx \frac{1}{N} \sum_{i=1}^{N} e^{\lambda_k r(\mathbf{J}_{i,\Lambda_k})}. \tag{5.28}$$

λ_k is the parameter (Lagrange multiplier) to satisfy constraint in eqn (5.26),

$$E_p[r_k] \approx \frac{1}{N} \sum_{i=1}^{N} \left[r(\mathbf{J}_{i,\Lambda_k}) e^{\lambda_k r(\mathbf{J}_{i,\Lambda_k})} \right] \frac{1}{z_k} = \bar{r}_k^+. \tag{5.29}$$

In computation, we can look up $\bar{r}_k^+$ in the table to find the best λ_k. The importance sampling is a good estimation in calculating λ_k and z_k because in our model r is one dimensional.

By recursion, we have a *factorized* log-linear form,

$$p_K(\mathbf{I}) = q(\mathbf{I}) \prod_{j=1}^{K} \left[\frac{1}{z_j} \exp\{\lambda_j r_j(\mathbf{I}_{\Lambda_j})\} \right]. \tag{5.30}$$

5.4 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 Gradients (HoG) and Scale-Invariant Feature Transfer (SIFT) representations, which was widely adopted in computer vision as generic image features and object templates before deep neural networks.

6

2.1D Sketch and Layered Representation

In the previous chapter, primal sketch is a generic two-layer 2D representation in term 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. In the pioneering work, the resulting representation is called *the 2.1D sketch*, by Nitzberg and Mumford [142], to bridge low-level image segmentation and middle-level Marr's 2.5D sketch (to be introduced in the next chapter), and termed *layered image representation*, by Adelson and Wang [1, 194, 195], in image coding and motion analysis.

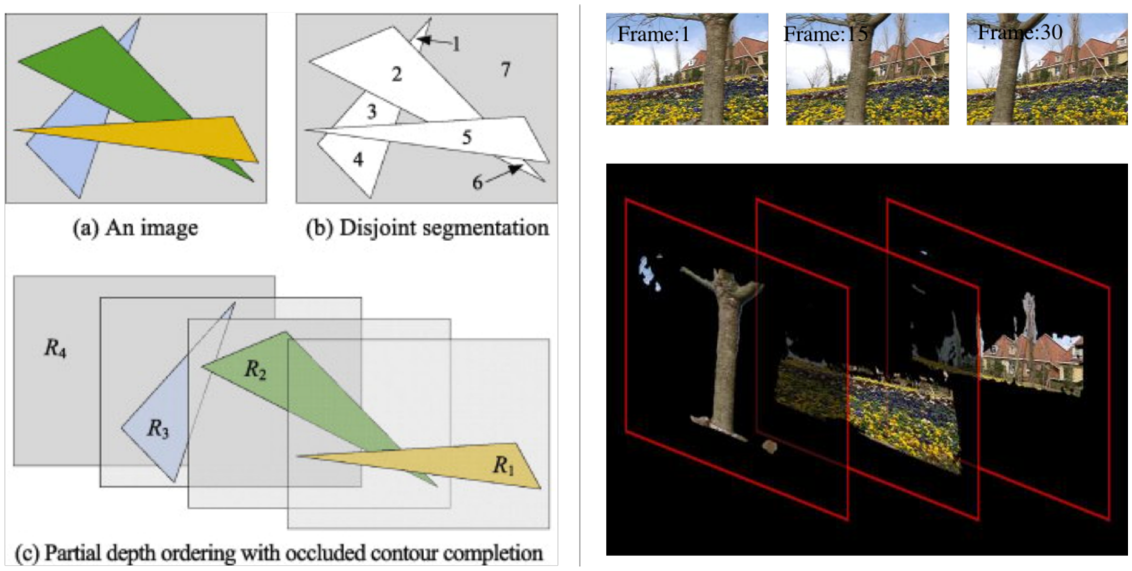

(a) An image
(b) Disjoint segmentation
1
2
7
3
5
4
6
(c) Partial depth ordering with occluded contour completion
R_4
R_2
R_3
R_1
Frame:1
Frame:15
Frame:30

Figure 6.1: Illustration of the 2.1D sketch and layered image representation. *Left:* a toy example of the 2.1D sketch, courtesy of [] which is reproduced for the illustrative example by Nitzberg and Mumford [142]. *Right:* a demo of the layered image representation in motion analysis, courtesy of John Y. A. Wang [194].

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 [124, 192], segmentation [45, 184], occlusion recovery [165, 198], contour illusory and completion [44, 82, 91, 152, 193], and foreground/background separation [153, 177].

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.

6.1 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 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 I, to its 2D representation, denoted by W_{2D} and to the 2.1D sketch, denoted by $W_{2.1D}$. We have,

$$I \Rightarrow \quad W_{2D} \Rightarrow \quad W_{2.1D}. \tag{6.1}$$

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,

$$W_{2D} = \{n, (R_i, l_i, \theta_i)_{i=1}^{n}\}, \tag{6.2}$$

which consists of n regions, R_i's each of which is represented by a region model indexed by l_i in a predefined model family with parameters θ_i fitted.

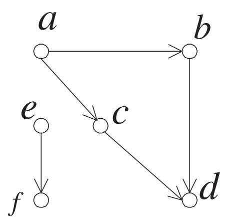


Figure 6.2: A Hasse diagram for a partial order relation

$W_{2.1D}$ represents a set of surfaces, their partial occluding order and contours used to complete regions in different surface if had. A surface consists of one 2D region or more than one 2D regions with completed contours if had. Any 2D region in W_{2D} belongs to one and only one surface in the layered representation. We have,

$$W_{2.1D} = \{m, \{S_i\}_{i=1}^{m}, \mathcal{PR}\} \tag{6.3}$$

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* [174], $\mathcal{PR} = \langle A, \preceq \rangle$. $b \preceq c$ 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 6.2 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,

$$S'_i = S_i \setminus \cup_{S_i \preceq S_j} S_j. \tag{6.4}$$

Then, the recovered curve(s) in the surface S_i can be defined by,

$$C_i = \partial S_i \setminus \partial S'_i. \tag{6.5}$$

Some C_i's may be empty. For example, the front most surface do 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 much larger search space and usually entails more powerful search algorithms such as DDMCMC [186] based ones.

6.2 The Variational Formulation by Nitzberg and Mumford

In their seminal work [142], Nitzberg and Mumford proposed a variational formulation for inferring the 2.1D sketch extended from the Mumford-Shah energy functional [136]. This belongs to the energy minimization framework and we briefly introduce the formulation in this section.

6.2.1 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 that 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 Γ 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 Γ segments the image domain into a finite set of disjoint regions, denoted by $R_1, \cdots, R_n$, 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, Γ:

$$E_{M-S}(f, \Gamma|I) = \mu^2 \int_{\mathcal{D}} (f - I)^2 dx + \int_{\mathcal{D}\backslash\Gamma} || \bigtriangledown f||^2 dx + v \int_{\Gamma} ds \tag{6.6}$$

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 a 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:

$$E_{2.1D}(\{S_i\}, \mathcal{PR}|I) = \sum_{i=1}^{n} \{\mu^2 \int_{S'_i} (I - m_i) + \epsilon \int_{S_i} dx + \int_{\partial S_i \setminus \partial\mathcal{D}} \phi(\kappa) ds\} \tag{6.7}$$

where m_i is the mean of the image I on S'_i and κ is the curvature of the boundary ∂S_i. The function $\phi : \mathbf{R} \to \mathbf{R}$ is defined by,

$$\phi(\kappa) = \begin{cases} v + \alpha\kappa^2, & \text{for } |\kappa| < \beta/\alpha \\ v + \beta|\kappa|, & \text{for } |\kappa| \geq \beta/\alpha \end{cases} \tag{6.8}$$

The functional above can only address inferring the 2.1D sketch from simple input images, which is reasonable assumption in the early day of computer vision. First, the model does not allow self-overlapping “woven" surfaces such as a garden hose would project, nor folded surfaces such as produced by an image of a sleeve whose edge disappears around the back of an arm. Second, the model is piece-wise 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.

6.2.2 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 ∂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:

$$E(C_i|S_i, P_0, t_0, P_1, t_1) = \int_{C_i} (v + \alpha\kappa^2) ds \tag{6.9}$$

Computationally, the simplest way to solve the Elastic seems to be hill-climbing. It starts with a convenient chain $x_0 = P_0, x_1, \cdot, x_N = P_1$ of points for which $x_1 - x_0 \propto t_0$ and $x_{N-1} - x_N \propto t_1$. X_i's ($i = 1, \cdots, N-1$) are computed by letting them evolve to decrease the Elastica functional.

6.3 The Mixed Markov Random Field Formulation

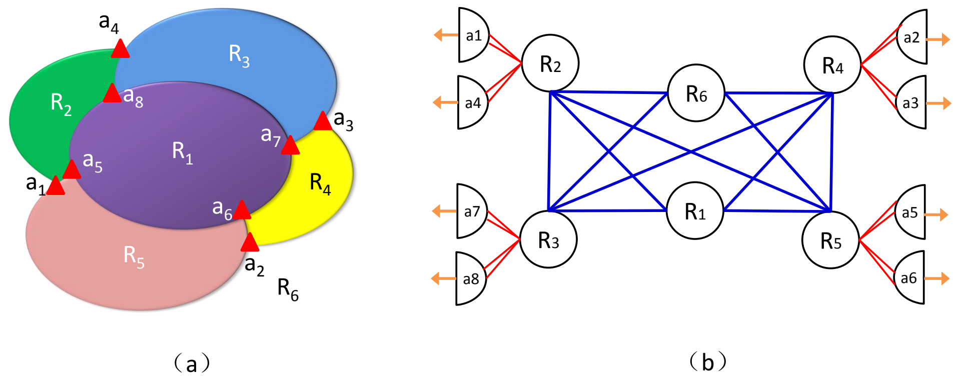


Figure 6.3: A simple example of the 2.1D sketch and the mixed MRF modeling. (a) is a set of 2D regions to be layered, (b) is the initial reconfigurable graph with mixed MRF including two types of nodes, region nodes and terminator nodes. See text for details.

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 6.3, 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.

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 6.1. 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 its 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 [49], [50] 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. 6.1, 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 $R_1, R_2, \cdots, R_6$ in Figure 6.3 (a)) and a set of terminators (see $a_1, a_2, \cdots, a_8$) 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 6.3 (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 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 [49].

6.3.1 Definition of W_{2D} and $W_{2.1D}$

For the clarity of 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:

$$W_{2D} = (V_R, V_T) \tag{6.10}$$

$$V_R = (R_1, R_2, ..., R_N) \tag{6.11}$$

$$V_T = (T_1, T_2, ..., T_M) \tag{6.12}$$

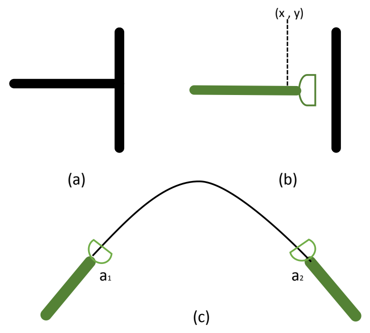


Figure 6.4: The open bounds or address variable. A T-junction is shown in (a), it is broken into a terminator in (b) as open bound described as an address variable including location (x, y), orientation of the terminator and appearance information inherited from the region it belongs to; (c) is the Elastica computed using two terminators in the process of contour completion.

V_R is the regions set, which can be many atomic regions or composed little image patches. V_T consists of those junctions selected from the detected 3-degree-junctions [199] 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. 6.4 (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, length, etc. 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 $B(R_i) \forall i \leq N$.

$$V_B = (a_1, a_2, ..., a_M) \tag{6.13}$$

$$A_B = (A_{a1}, A_{a2}, ..., A_{aM}), a_i \in V_B, \forall i \leq M \tag{6.14}$$

$$B(R_i) = (a_{i1}, a_{i2}, ..., a_{im}), a_{ij} \in V_B, \forall i \leq N, \forall j \leq m \tag{6.15}$$

Generative models of regions. In this chapter, we adopt generative models for each region. Let $R \subset \Lambda$ denote a 2D region and 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 η modeled by a non-parametric histogram H. The model can be learned off-line and more sophisticated models can easily be added into the algorithm.

$$J(x, y, \theta) = \mu, I_R(x, y) = J(x, y, ; \theta) + \eta, \eta \sim H \tag{6.16}$$

With a slight abuse of notation, we denote the parameters used in a region by $\theta = (\mu, H)$, here μ is the mean of a region R or a connected component CP. The likelihood probability is

$$p(I_R|R, \theta) \propto \prod_{(x,y \in R)} H(I_R(x, y) - J(x, y, \theta)) \tag{6.17}$$

The prior for a region R assumes short boundary length ∂R. This is to encourage smoothness, and a compact area $\|R\|$,

$$p(R) \propto \exp\{-\gamma_r |R|^\rho - \frac{1}{2}\lambda|\partial R|\} \tag{6.18}$$

where ρ and λ are fixed constants and γ 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 $A_{ai} = (x, y; ori, \kappa, pf)$, where (x, y) is location, ori the orientation of the terminator, κ 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 Γ^*, is decided by minimizing the Elastica cost function in a contour space Ω_Γ [142], [100], [136], [85]

$$\begin{aligned}\Gamma^* &= \arg\min_{\Gamma\in\Omega_\Gamma} E(\Gamma; a_i, a_j) \\ &= \arg\min_{\Gamma\in\Omega_\Gamma} \int_\Gamma [(\nu_1 + \alpha_1\kappa_1^2) + (\nu_2 + \alpha_2\kappa_2^2)]ds\end{aligned} \tag{6.19}$$

where κ_1 is the curvature of a_1, κ_2 is the curvature of a_2, ν_1 and ν_2 are constants, and α_1 and α_2 are scalable coefficients. The parameters ν_1, ν_2, α_1 and α_2 are learned from an image data set [213].

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.

$$W_{2.1D} = (X, Y; S_f, C_t) \tag{6.20}$$

$$X = (x_{R_1}, x_{R_2}, ..., x_{R_N}) \tag{6.21}$$

$$Y = (y_{a_1}, y_{a_2}, ..., y_{a_M}) \tag{6.22}$$

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 indicating 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 include 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 the Elastica based on the results of assignments.

6.3.2 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 6.3 (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 $x_{xa} = y_{ya}, \forall a \in C \bigcap V_B$. Thus a mixed potential function λ_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: $P(I) = \alpha \prod_{C \in \mathbb{C}} \lambda_C(I_C, I_{I_C})$, 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 originally established the 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 [49] or SW-cuts [6] according to A.Fridman's proof in [49].

Given a region R_i, its neighborhood is:

$$N(R_i) = N_{adjacency}(R_i) \cup N_{pointer}(R_i) \tag{6.23}$$

where $N_{adjacency}(R_i) = \{R_j : \forall R_j \in V_R \text{ is adjacent to } R_i\}$ 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, $N_{pointer}(R_i) = \{R_j : B(R_j) \text{ and } B(R_i) \text{ are connected}\}$. Then E_R is:

$$E_R = \{\langle R_i, R_j \rangle, \forall R_i \in V_R \text{ and } R_j \in N(R_i)\} \tag{6.24}$$

Given an open bond $a_i \in V_B$, its neighborhood is:

$$N(a_i) = \{a_j : a_j \text{ and } a_i \text{ are in the same layer}\} \tag{6.25}$$

Hence $E_B = \langle a_i, a_j \rangle, \forall a_i \in V_T$ and $a_j \in N(a_i)$. At the initial stage, all the open bonds are open.

6.3.3 Bayesian formulation

In the Bayesian framework, the three-layer model in the Eqn. 6.1 is described as

$$p(I, W_{2D}, W_{2.1D}) = p(I|W_{2D}, W_{2.1D})p(W_{2D}|W_{2.1D})p(W_{2.1D}) \tag{6.26}$$

where $p(I|W_{2D}, W_{2.1D})$ is the likelihood model stated in Eq.(8). We want to maximize the posterior joint probability of $W_{2.1D}$ given W_{2D} in a solution space $\Omega_{W2.1D}$

$$\begin{aligned} W^*_{2.1D} &= \arg \max_{\Omega_{W_{2.1D}}} p(W_{2.1D}|W_{2D}; I) \\ &= \arg \max_{\Omega_{W_{2.1D}}} p(W_{2.1D}|W_{2D})p(W_{2.1D}) \end{aligned} \tag{6.27}$$

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

$$\pi_n = (V_1, V_2, ..., V_n), \bigcup_{i=1}^{n} V_i = V, V_i \bigcap V_j =, \forall i \neq j \tag{6.28}$$

Each subset $V_i, i = 1, 2, ..., n$ (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, Ω_{lr} the set of types of region models, $\Omega_{\theta r}$ the model parameter family, Ω_{lc} the set of types of Elastica, and $\Omega_{\theta c}$ the Elastica parameter space. Thus, the solution space for $W_{2.1D}$ is

$$\Omega = \bigcup_{n=1}^{N} \{\Omega_{\pi_n} \times \Omega_{l_r}^n \times \Omega_{\theta_{r1}} \times ... \times \Omega_{\theta_{rn}} \times \Omega_{l_c}^n \times \Omega_{\theta_{c1}} \times ... \times \Omega_{\theta_{cn}}\} \tag{6.29}$$

This leads us to extend the Swendsen-Wang cuts algorithm [6] to perform the inference. In the formulation, the prior model is

$$p(W_{2.1D}) = p(K)p(X)p(Y) \prod_{i=1}^{N} p(R_i) \tag{6.30}$$

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.(6.18).

The likelihood model is

$$\begin{aligned} p(W_{2D}|W_{2.1D}) \propto & \prod_{i=1}^{K} \prod_{j=1}^{K(i)} \prod_{(x,y) \in s_j} H(I(x,y) - J(x,y;\theta_j)) \\ & \times \prod_{i=1}^{M} \exp\{-\lambda_0 E\} \end{aligned} \tag{6.31}$$

where $J(x, y; \theta_j)$ is defined in Eq.(6.16) and λ_0 is a scalable factor and also learned from the data set [213]. 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.

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 [6] 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 π and π_0 in terms of the Metropolis-Hastings method.

6.3.4 Experiments

We first demonstrate the inference algorithm using the Kanizsa image. Figure 6.5 shows the 2D representation and the graphical representation with mixed MRF. Figure **??** shows the inference procedure.

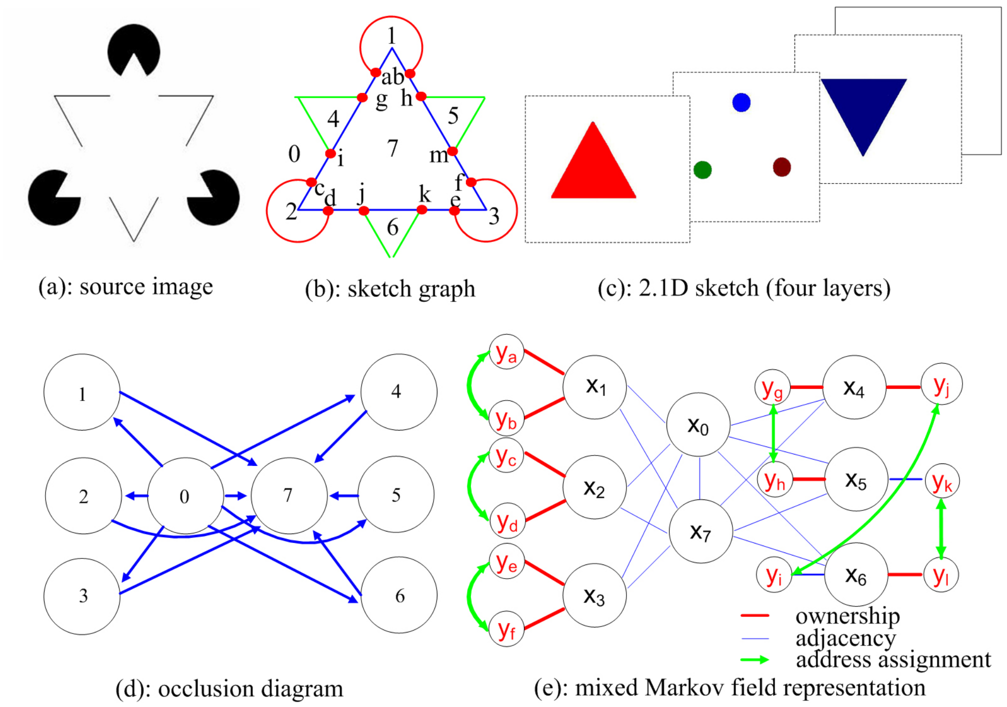


Figure 6.5: A running example of inferring the 2.1D sketch. (a) is the original image, Kanizsa figure. (b) is the sketch graph computed by the primal sketch model after interactively labeling. there are 8 atomic regions and 12 terminators broken form T-junctions. (c) is the 2.1D sketch. There are 4 layer and the contour completion is performed using Elastica rules. (d) is the Hasse diagram for partial order relation, such as $< 7, 1 >$ means region 7 occludes region 1. (e) is the graph representation with mixed Markov random field. Each big circle denotes a vertex of atomic region, each blue 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 red two-way arrows connect two terminators and each terminator is assigned another terminator's address variable.

6.4 The 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 6.6. Given an input image, our objective is to infer an unknown number 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 which contains a large number of subspaces of varying dimensions for the possible combinations of regions, curves, and curve groups.

6.4.1 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 6.7 and described as-follows.

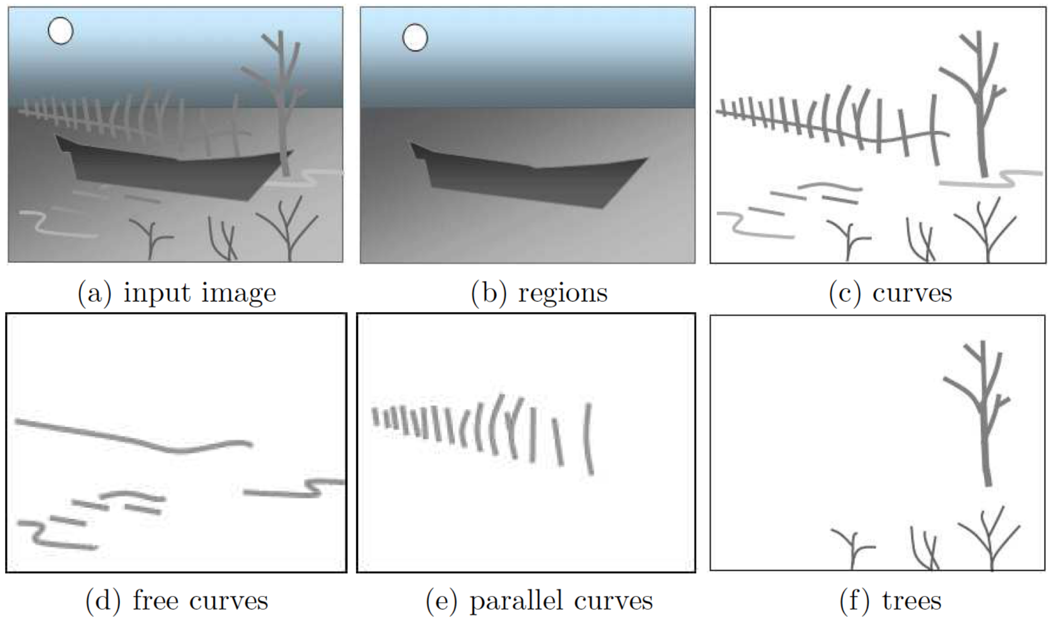


Figure 6.6: Illustration of the 2.1D sketch with layered representation of regions and curves. (a) is an input image which is decomposed into two layers – (b) a layer of regions and (c) a layer of curves. These curves are further divided into (d) free curves, (e) a parallel curve group for the fence, and (f) trees. Curves observe a partial order occlusion relation.

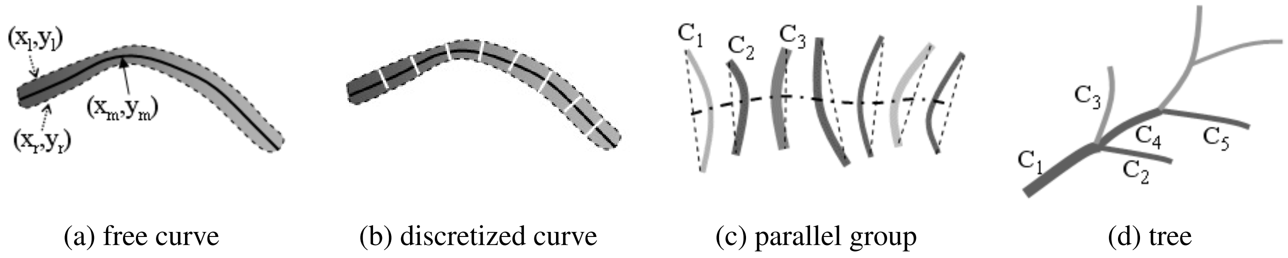


Figure 6.7: Representations of curves and curve groups. (a) A free curve in continuous representation. (b) A free curve is discretized into a chain of "bars". (c) Curves for a parallel group (d) Curves for a Markov tree.

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. (6.7.a) shows the boundaries in dashed lines.

$$\mathbf{c}_l(s) = \mathbf{c}_m(s) - w(s)\mathbf{n}(s), \quad \mathbf{c}_r(s) = \mathbf{c}_m(s) + w(s)\mathbf{n}(s), \tag{6.32}$$

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 $w << L$. 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 often exhibit globally smooth shading patterns, for example the curves in Figure 6.7. Thus we adopt a quadratic function for curve intensities,

$$\mathbf{J}(x, y; \theta_0) = ax^2 + bxy + cy^2 + dx + ey + f, \quad (x, y) \in \mathcal{D}(C), \tag{6.33}$$

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 [186]. Therefore, a free curve is described by the following variables in continuous representation

$$C = (L, \ \mathbf{c}_m(s)_{s=0}^{L}, \ w(s)_{s=0}^{L}, \ \ \theta_0, \sigma).$$

where σ 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 6.7.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.

$$p(C) \propto p(\mathcal{D}(C))p(\mathbf{c}(s))p(w(s)) \propto e^{-E(C)}. \tag{6.34}$$

The energy $E(C)$ is the sum of three terms

$$E(C) = \gamma_c|\mathcal{D}(C)|^{\rho} + \lambda L + E_o(w), \tag{6.35}$$

where ρ, λ are constants and are fixed in our experiments, and γ_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

$$p(\mathbf{I}_{\mathcal{D}(C)}|C) = \prod_{(x,y)\in\mathcal{D}(C)} \mathcal{N}(\mathbf{I}(x, y) - \mathbf{J}(x, y; \theta_0); \ \sigma^2). \tag{6.36}$$

2. Parallel curve groups. A parallel curve group consists of a number of nearly parallel curves as Fig. (6.7.c) shows. Each curve $C_i, i = 1, 2, ..., n$ 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 coding length and it is useful for perceiving an object, for example, a zebra. We denote a parallel curve group by

$$pg = (n, \{C_1, C_2, ..., C_n\}, \{\alpha_1, \alpha_2, ..., \alpha_n\}),$$

$\alpha_i \in \{1, ..., n\}$ 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

$$p(pg) \propto \exp\{-\lambda_0 n - \sum_{i=1}^{n} E(C_i) - \sum_{i=2}^{n} E_{pg}(C_i, C_{\alpha_i})\}. \tag{6.37}$$

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 θ_i of its associate line-segment, and length L_i of the line segment, curve average width (thickness) $\bar{w}_i$, and average intensity μ_i. $E_{pg}(C_i, C_{\alpha_i})$ measures the differences between these attributes.

3. Markov trees. Figure 6.7.d shows a number of curves in a Markov tree structure. We denote it by

$$T = (n, \{C_1, C_2, ..., C_n\}, \{\beta_1, \beta_2, ..., \beta_n\}).$$

$\beta_i \in \{1, ..., n\}$ is the index to the parent curve of C_i. Thus the prior probability is

$$p(T) \propto \exp\{-\lambda_0 n - \sum_{i=1}^{n} E(C_i) - \sum_{\alpha_i \neq \emptyset} E_T(C_i, C_{\beta_i})\}. \tag{6.38}$$

Again, $E(C_i)$ is inherited from the free curve. The term for C_i and its parent C_{α_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

$$\mathcal{D}(pg) = \cup_{i=1}^{n} \mathcal{D}(C_i), and\, \mathcal{D}(T) = \cup_{i=1}^{n} \mathcal{D}(C_i). \tag{6.39}$$

It also inherits the intensity function $\mathbf{J}(x, y; \theta_i)$ from each free curve $C_i, i = 1, 2, ..., n$. In summary, the intensity models for C, pg, T are all generative for image $\mathbf{I}$ as

$$\mathbf{I}(x, y) = \mathbf{J}(x, y; \theta) + \mathcal{N}(0; \sigma^2), \ \ (x, y) \in \mathcal{D}(C_i),\ \mathcal{D}(pg),\ \text{or}\ \mathcal{D}(T). \tag{6.40}$$

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 [186]. We denote a 2D region by $R \subset D$ and $\mathbf{I}_R$ the intensities inside R.

The first model assumes constant intensity with additive noise modeled by a non-parametric histogram $\mathcal{H}$.

$$\mathbf{J}(x, y; 1, \theta) = \mu, \ \ \mathbf{I}(x, y) = \mathbf{J}(x, y) + \eta, \ \ \eta \sim \mathcal{H}, \ \ (x, y) \in R.$$

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.

$$\mathbf{J}(x, y; 2, \theta) = B'(x) M B(y), \ \ \mathbf{I}(x, y) = \mathbf{J}(x, y; \theta_2) + \eta, \ \ \eta \sim \mathcal{H}, \ \ (x, y) \in R.$$

where $B(x) = ((1-x)^3, 3x(1-x)^2, 3x^2(1-x), x^3)$ is the basis and M is a 4×4 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 [186] where we compare different models for different types of images.

The likelihood probability is

$$p(\mathbf{I}_R | R, \theta) \propto \prod_{(x,y) \in \mathcal{D}(R)} \mathcal{H}(\mathbf{I}(x, y) - \mathbf{J}(x, y; \ell, \theta)), \quad \ell \in \{1, 2\}. \tag{6.41}$$

The prior for a region R assumes short boundary length ∂R (smoothness) and compact area $|\mathcal{D}(R)|$,

$$p(R) \propto \exp\{-\gamma_r |\mathcal{D}(R)|^\rho - \frac{1}{2}\lambda |\partial R|\}, \tag{6.42}$$

where ρ and λ are constants that are fixed for all the experiments in this section, and γ_r is a scale factor that can be adjusted to control the number of regions in the segmentation.

6.4.2 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

$$W = (W^r, W^c, W^{pg}, W^t, \mathcal{PR}).$$

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 θ_i for its intensity model

$$W^r = (K^r, \{(R_i, \ell_i, \theta_i) : i = 1, 2, ..., K^r\}).$$

Similarly, we have $W^c = (K^c, C_1, ..., C_{K^c})$, $W^{pg} = (K^{pg}, pg_1, pg_2, ..., pg_{K^{pg}})$, and $W^t = (K^t, T_1, T_2, ..., T_{K^t})$. 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 Ω.

$$W^* = \arg\max_{W \in \Omega} p(\mathbf{I}|W)p(W).$$

By assuming mutual independence between W^r, W^c, W^{pg}, W^t we have the prior model

$$p(W) = \Big(p(K^r)\prod_{i=1}^{K^r} p(R_i)\Big)\Big(p(K^c)\prod_{i=1}^{K^c} p(C_i)\Big)\Big(p(K^{pg})\prod_{i=1}^{K^{pg}} p(pg_i)\Big)\Big(p(K^t)\prod_{i=1}^{K^t} p(T_i)\Big). \quad (6.43)$$

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 eqns (6.36) and (6.41).

$$p(\mathbf{I}|W) = \prod_{i=1}^{K^r} \prod_{(x,y)\in D_{R_i}} \mathcal{H}((\mathbf{I}(x, y) - \mathbf{J}(x, y; \ell_i, \theta_i)) \cdot \prod_{j=1}^{N} \prod_{(x,y)\in D_{C_j}} \mathcal{N}((\mathbf{I}(x, y) - \mathbf{J}(x, y; \theta_j); \sigma_j^2). \quad (6.44)$$

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. More details are referred to []. 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 states 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

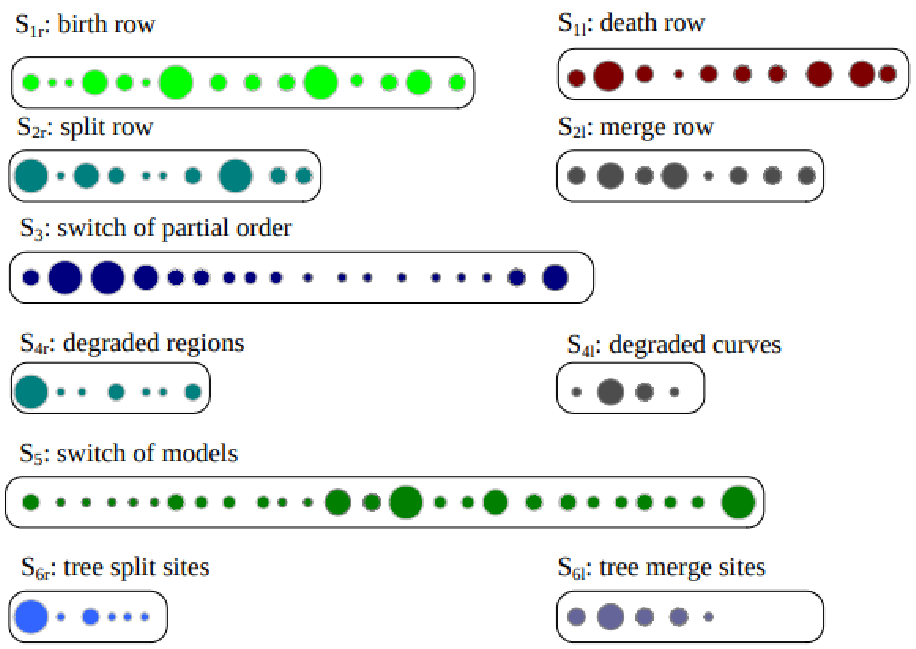


Figure 6.8: The 6 simple jumps maintain 10 sets of "particles" whose sizes illustrate their weights. The sets are updated and re-weighted in each jump steps, and they encode the proposal probabilities in a non-parametric representation.

sampler. The proposal is then accepted in a Metropolis-Hastings step, hence its name, the *Metropolized Gibbs Sampler* (MGS [121]).

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. (6.8) 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.

6.4.3 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 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. (6.9). 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 $\mathbf{I}^r_{\text{syn}} \sim p(\mathbf{I}|W^r)$, the curves $\mathbf{I}^c_{\text{syn}} \sim p(\mathbf{I}|W^c)$, and the overall synthesis $\mathbf{I}_{\text{syn}}$ by occluding $\mathbf{I}^c_{\text{syn}}$ on $\mathbf{I}^r_{\text{syn}}$.

We construct synthesis image to verify how an input image is represented in W^*. In the experiment, two parameters in the prior models are adjustable: (1) γ_r in equation (6.42), and (2) γ_c in equation (6.35). 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.

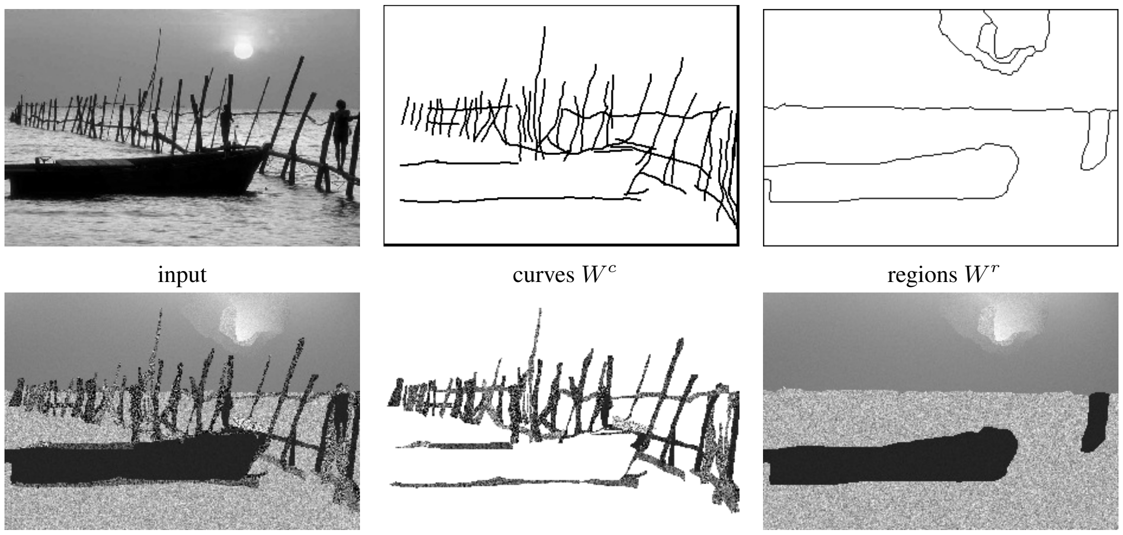

input
curves W^c
regions W^r
synthesis
synthesis $\mathbf{I}^c_{syn} \sim$ by W^c
synthesis $\mathbf{I}^r_{syn} \sim$ by W^r

Figure 6.9: Experiment: parsing images into regions and free curves.

Experiment B: assuming regions, curves and parallel groups, and trees

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 (6.10) 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.

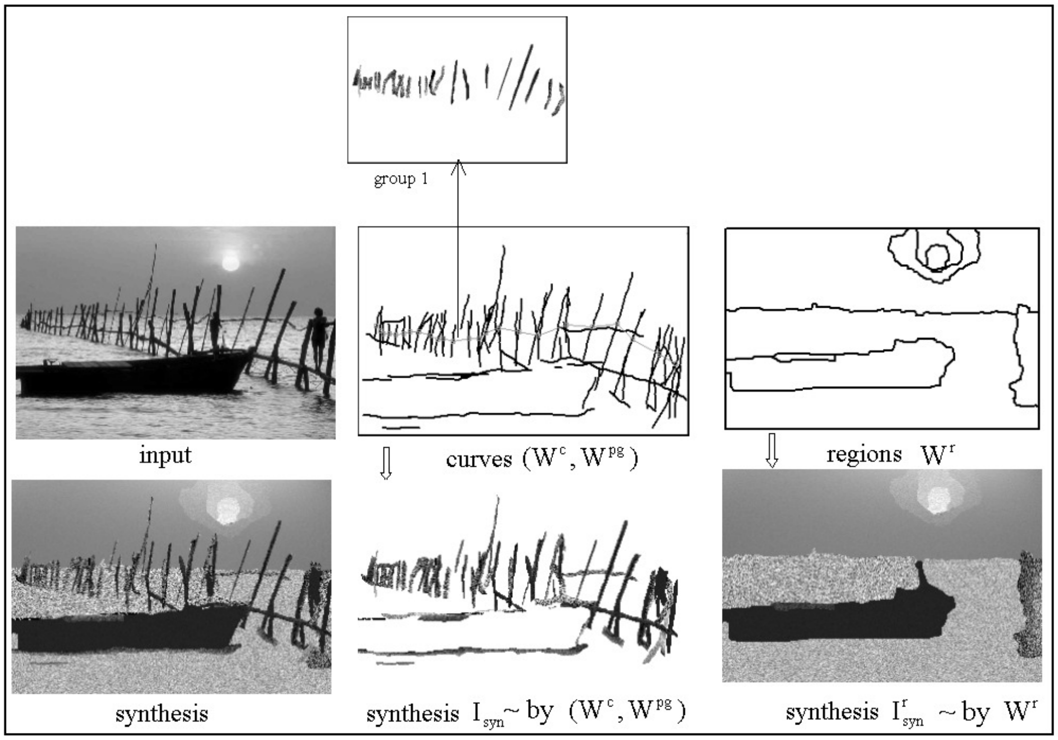

group 1
input
curves (W^c, W^{pg})
regions W^r
synthesis
synthesis $I_{syn} \sim$ by (W^c, W^{pg})
synthesis $I^r_{syn} \sim$ by W^r

Figure 6.10: Experiment B1: parsing an image into regions, curves, and parallel curve groups.

7

2.5D Sketch and Depth Maps

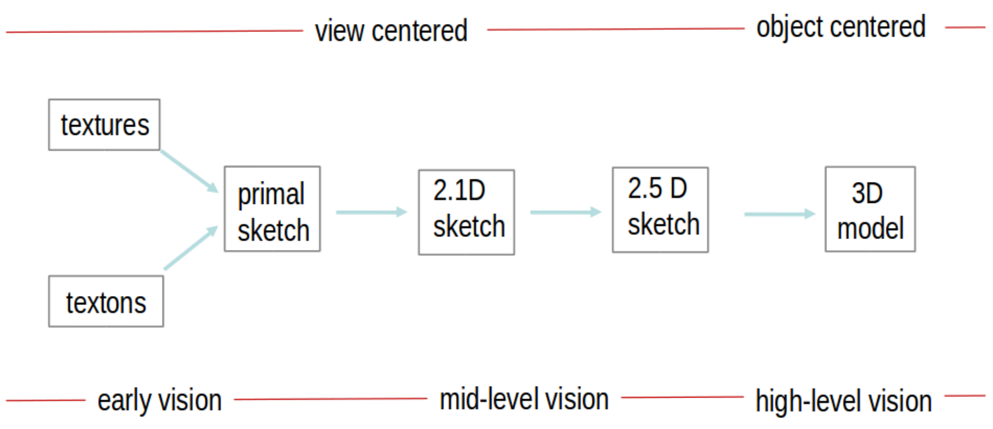


Figure 7.1: Representation in Marr's paradigm. The 2.1D sketch was proposed by Nitzberg and Mumford [142]

.

In previous chapter, we discuss about the primal sketch and 2.1D sketch. Primal sketch is a generic two-layer 2D representation and 2.1D sketch aims to compute layered representation of an input image. In this chapter,we will discuss the idea of the 2.5D sketch. The idea of the 2.5D sketch first appeared in Marr and Nishihara [129], it provides a viewer-centered representation of the visible surfaces. As shown in Figure 7.1, the construction of the 2.5D sketch is considered to be a significant part in mid-level vision, making the last step before a surface's interpretation and the end, perhaps, of pure perception.

7.1 Marr's Definition

We will first describe the original definition for a viewer-centered representation that uses surface primitives of small size as described in [128]. It includes a representation of contours of surface discontinuity, and it has engough 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. Surface discontinuities may be represented by oriented line elements. Surface orientation may be represented as a vector (p, q) in two-dimensional space, which is equivalent to covering the image with needles. The length of each needle defines the slant (or dip) of the surface at the

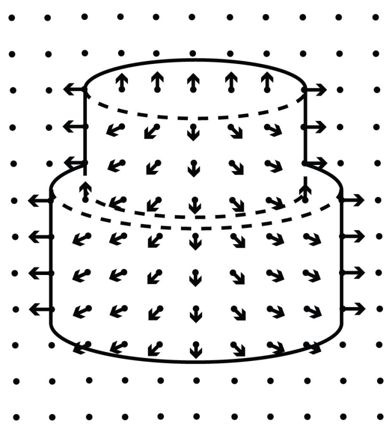


Figure 7.2: Example of a 2.5D Sketch described in [128]. The surface orientation is again represented by arrows, as explained in the text. Occluding contours are shown with full lines, and surface orientaion discontinuities with dotted lines. Depth is not shown in the figure, though it is thought that rough depth is available in the representation.

point, so that zero length corrersponds to a surface that is perpendicular to the vector from the viewer to that point, and the length of the needle increases as the surface slants away from the viewer. The orientation of the needle defines the tilt, that is, the direction of the surface's slant. Figure 7.2 illustrates this representation; it is like having a gradient space at each point in the visual field.

In principle, the relation between depth and surface orientation is straightforward—one is simply the integral of the other, taken over regions bounded by surface discontinuities. It is therefore 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 about the approaches to generate 2.5D sketch in following sections. It includes two generative methods which generate 2.5D sketch from primal sketch in applications of shape from stereo and shape from shading, and one recent discriminative approach in deep learning.

7.2 2.5D Sketch from Primal Sketch — Shape from Stereo

In this part, we present a two-level generative model that incorporates generic visual knowledge for dense stereo reconstruction. 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. We propose an MCMC algorithm that simultaneously infers the 3D primitive types and parameters and estimates the depth (2.5D sketch) of the scene. Our experiments show that this representation can infer the depth map with sharp boundaries and junctions for textureless images, curve objects, and free-form shapes.

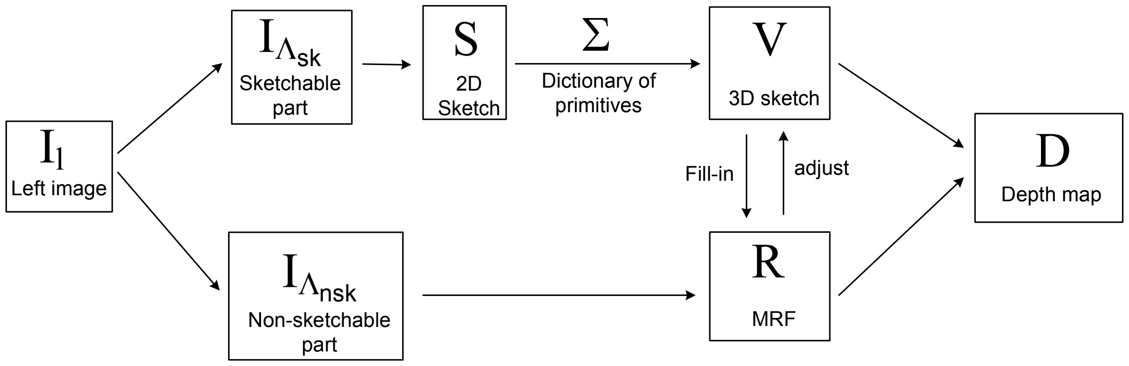


Figure 7.3: The flow diagram of the algorithm

.

The overall dataflow of the algorithm is illustrated in Figure 7.3. Given a stereo pair of images, we first compute a primal sketch representation [70] 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.

We then study the 3D structures for these points, line segements, and junctions and develop 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 normal 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 is summarized in a dictionary of junctions. Figure 7.6 and 7.7 shows 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.

We adopt 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 [50], 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.

Our model 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 a broad view, in this section we revisit the 2.5D sketch envisioned by Marr in [128]. It is organized as follows. We first present the representation, dictionary of primitives and junctions and probabilistic formulation in section 7.2.1, and then explain the algorithm details in section 7.2.2. We show results in section 7.2.3.

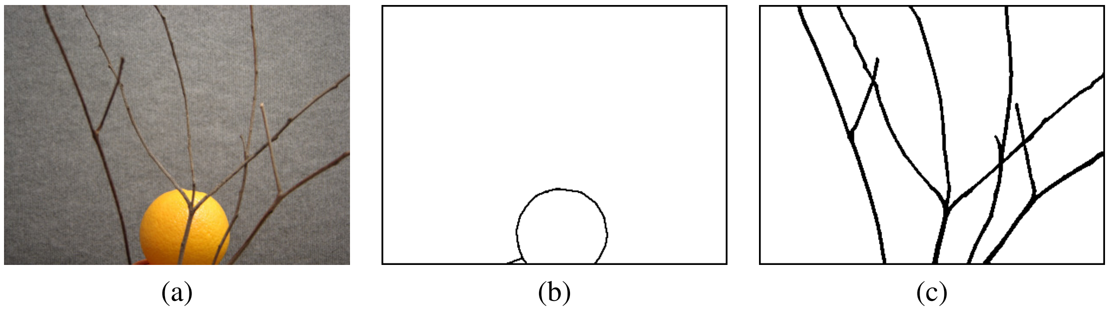


Figure 7.4: Our algorithm starts from a two layer sketch representation. (a) input image, (b) region layer, (c) curve layer.

7.2.1 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 Λ be the pixel lattice.

We assume the disparity map D is generally continuous and differentiable, with the exception of a number of curves Λ_{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. We 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 7.4. The curve layer is assumed to occlude the region layer. These sketches can be obtained as in [70, 187]. The sketch edges are approximated with line segments $S = \{s_i, i = 1, .., n_e\}$. 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 respectively on the right of s_i, as shown in Figure 7.5, left. Each curve segment $s_j \in S_C$ is assigned a *curve region* r_j along the segment, of width equal to the width of the curve, as shown in Figure 7.5, right. Denote the pixels covered by the edge and curve regions by Λ_R, Λ_C respectively.

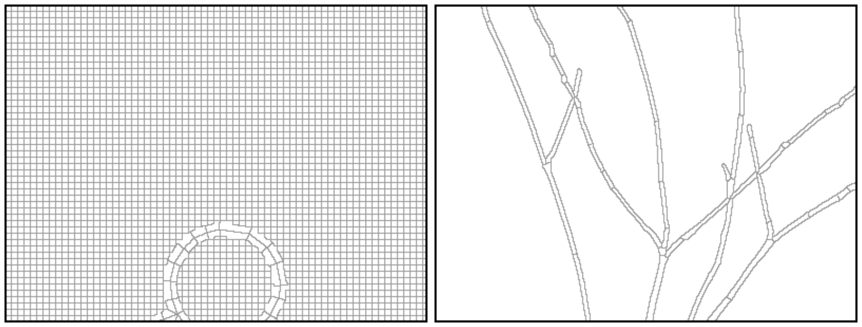


Figure 7.5: Division of the image in Figure 7.4 into sketch primitives and 6x6 pixel square regions. Region layer (left) and curve layer (right).

Because away from the places of discontinuity, the surfaces are in general very smooth and to reduce the dimensionality of the problem, the pixels of $\Lambda \setminus \Lambda_R$ are grouped into *square regions* of size 6×6 pixels, by intersecting $\Lambda \setminus \Lambda_R$ with a 6×6 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 Λ is divided into atomic regions that either are along the sketch S_C, or are on the rectangular grid, as shown in Figure 7.5. 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 in Figure 7.6. Depending on the type of segments they originated from, there are *boundary primitives* and *curve primitives*.

B. A dictionary of primitives. Let

$$V_1 = \{\pi_i = (s_i, [l_i, o_i^l], r_i, o_i^r, t_i, p_i, d_i, w_i[, f_i]), i = 1, .., n_e\} \tag{7.1}$$

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 $t_i = t(\pi_i) \in \{1, .., 8\}$ 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 $\{1, .., 6\}$ while curve segments $s_i \in S_C$ can only be assigned types from $\{1, 7, 8\}$. These types are illustrated in Figure 7.6.
   - 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.
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).

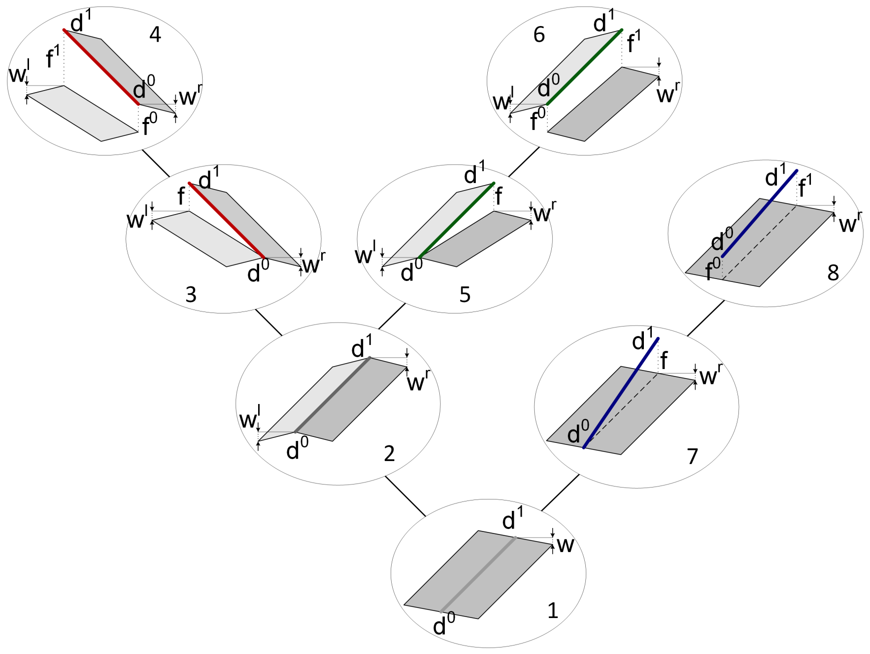


Figure 7.6: Each sketch segment is augmented to a primitive from the following dictionary, ordered by generality.

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

$$c(r_i, d) = \begin{cases} 0 & \text{if } r_i \text{ intersects the curve sketch } S_C \\ \sum_{v \in r_i} |I_l(v) - I_r(v - d_v(\pi_i))| & \text{if } o_i^r = 1 \\ \alpha & \text{else} \end{cases} \tag{7.2}$$

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 $w = 0$. Then the matching cost of the primitive π_i is

$$c(\pi_i) = c(r_i, [l_i], t_i, d_i, [f_i]) = \begin{cases} c(r_i, d_i) & \text{if } t_i = 7, 8, 1(\text{curve}) \\ c(l_i, d_i) + c(r_i, d_i) & \text{if } t_i = 2, 1(\text{region}) \\ c(l_i, f_i) + c(r_i, d_i) & \text{if } t_i = 3, 4 \\ c(l_i, d_i) + c(r_i, f_i) & \text{if } t_i = 5, 6 \end{cases} \tag{7.3}$$

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 [166] we will have certain types of possible junctions depending on the degree (number of primitives) of the junction, as mentioned below and illustrated in Figure 7.7.

- 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.

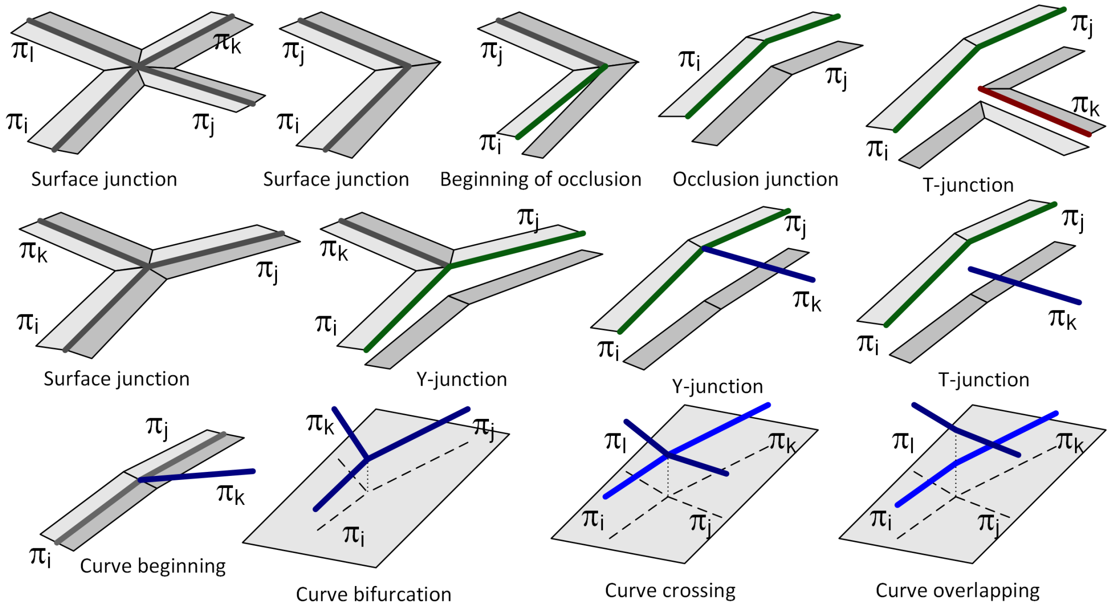


Figure 7.7: These are the main types of junctions between boundary and curve primitives.

- we assume there are 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 $J = \{\phi_i = (t, k, \pi_{i_1}, ..., \pi_{i_k}), \pi_{i_1}, ..., \pi_{i_k} \in V_1, i = 1, ..., n_J\}$ 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 $\pi_{i_1}, ..., \pi_{i_k}$ 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 $\pi_{i_1}, ..., \pi_{i_k}$ 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

$$P(\phi) \propto P(\pi_{i_1}, ..., \pi_{i_k}|t, \phi^{2D})P(\phi^{2D}, t) = P(\phi^{3D}|t, \phi^{2D})P(t|\phi^{2D}) \tag{7.4}$$

since the 2d geometry ϕ^{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:

$$\begin{aligned} p_c(\pi_i, \pi_j) &= \frac{1}{Z_c}\exp(-\beta_c|d_i^\phi - d_j^\phi|^2) \\ p_s(\pi_i, \pi_j) &= \frac{1}{Z_s}\exp[-\beta_c|d_i^\phi - d_j^\phi|^2 - \beta_s(|d_i^{\bar\phi} - 2d_i^\phi + d_j^{\bar\phi}|^2 - |d_i^{\bar\phi} - 2d_j^\phi + d_j^{\bar\phi}|^2)] \end{aligned} \tag{7.5}$$

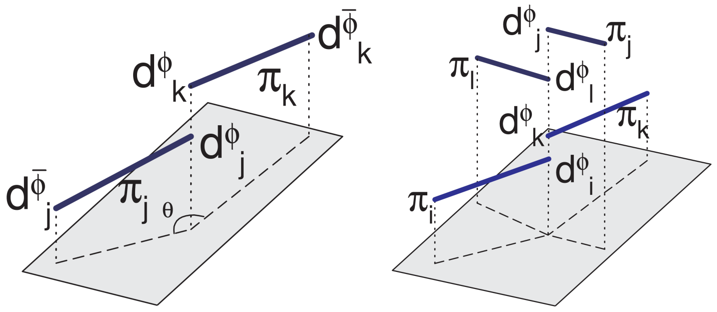


Figure 7.8: Left: The prior of the junction between two boundary or curve primitives depends on the angle θ between the primitives. Right: The prior of the curve overlapping junction encourages continuity of each pair of opposite curves.

where $d_i = (d_i^\phi, d_i^{\bar\phi})$ is the disparity of the primitive π_i, with d_j^ϕ being the disparity at the junction ϕ 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.

$$P(\phi^{3D}|t, \phi^{2D}) = \frac{1}{Z_1} \prod_{\pi_j, \pi_k \in \phi} p_c(\pi_j, \pi_k) \tag{7.6}$$

2) The prior of junctions of two boundary or two curve primitives depends on the angle θ between the primitives at the junction.

$$P(\phi^{3D}|t, \phi^{2D}) = \frac{1}{Z_2} \begin{cases} p_c(\pi_j, \pi_k) & \text{if } |\theta - \pi| > \pi/6 \\ p_s(\pi_j, \pi_k) & \text{else} \end{cases} \tag{7.7}$$

as shown in Figure 7.8, 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.

$$P(\phi^{3D}|t, \phi^{2D}) = p_s(\pi_i, \pi_k) p_s(\pi_j, \pi_l) \tag{7.8}$$

as shown in Figure 7.8, 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 π_i, π_j (refer to Figure 7.7) to have a good continuation as in case 2).

$$P(\phi^{3D}|t, \phi^{2D}) = \frac{1}{Z_4} p_s(\pi_i, \pi_j) \prod_{\pi_u, \pi_v \in \phi, \{u,v\} \neq \{i,j\}} p_c(\pi_u, \pi_v) \tag{7.9}$$

5) For the T-junctions, the prior encourages continuity of the occluding edge.

$$P(\phi^{3D}|t, \phi^{2D}) = p_s(\pi_i, \pi_j) \tag{7.10}$$

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

$$\psi_{\pi_i}(d,[f]) = \exp(-c(r_i,[l_i],t_i,d,[f])/10) \tag{7.11}$$

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

$$\xi(\pi_i) = \frac{\min_d c(l_i,d)}{|l_i|} - \frac{\min_d[c(l_i,d)+c(r_i,d)]}{|l_i|+|r_i|} \tag{7.12}$$

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 ξ 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.

$$L_\pi(t) = \begin{cases} 60e^{-\xi^2/2} \text{ if } \ t=1,2 \\ 4.4e^{-(\xi+1.18)^2/1.42} + 3.67e^{-(\xi+8.21)^2/6} \text{ if } \ t=3,4 \\ 9.18e^{-(\xi-0.58)^2/0.87} + 3.06e^{-(\xi-7.36)^2/7.5} \text{ if } \ t=5,6 \end{cases} \tag{7.13}$$

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 $\phi = \{\pi_1, ..., \pi_k\}$:

$$L_\phi(t) = P(\phi)L_{\pi_1}(t_1)...L_{\pi_k}(t_k) \tag{7.14}$$

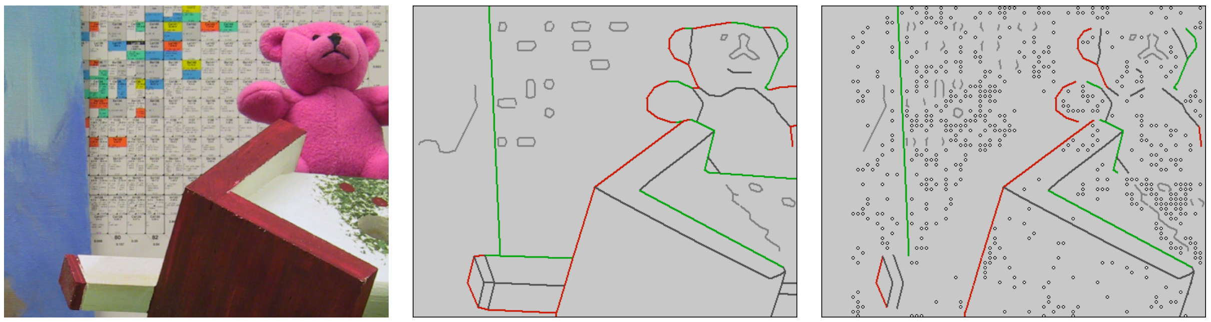


Figure 7.9: Left image of a stereo sequence, the graph labeling and the control points (point and boundary primitives) of the thin plate spline.

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

$$\rho_r(d) \propto \exp(-\sum_{v\in r} |I_l(v) - I_r(v-d))|/10) \tag{7.15}$$

to all possible disparities $d \in [d_{\min}, d_{\max}]$. Then the square regions

$$R = \{r_i = (d_i, o_i, p_i, \mu_i, \sigma_i^2), i = 1, .., n_r\} \tag{7.16}$$

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 μ_i and variance σ_i^2 of the saliency map ρ_{r_i}.

Following [9], all regions (edge regions, curve regions, and square regions) will have their occlusion label deterministically assigned based on the disparities of the boundary and curve primitives. For example, for an occlusion primitive π_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 π_i will be labeled as occluded.

The matching cost for each region $r_i \in R$ is

$$c(r_i) = \begin{cases} \alpha & \text{if } o_i = 0 \\ \sum_{v \in r_i} |I_l(v) - I_r(v - d_i))| & \text{if } o_i = 1 \end{cases} \tag{7.17}$$

The set of point primitives is denoted by

$$V_0 = \{r_i \in R, s_i = 1\}. \tag{7.18}$$

In Figure 7.9 are shown the labeled graph, i.e. primitive types (middle), and the point and boundary primitives that act as control points for the Λ_{nsk} part (right). The depth and disparity maps obtained this way are shown in Figure 7.13. 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 7.13.

E. Bayesian formulation. We formulate our model using the Bayes rule:

$$P(V_1, R|I_l, I_r) = P(I_l|I_r, V_1, R)P(R - V_0|V_0, V_1)P(V_0, V_1) \tag{7.19}$$

The likelihood $P(I_l|I_r, V_1, R)$ is expressed in terms of the likelihood $L_{\pi_i}(t_i)$ and matching cost $c(r_j)$ of the sketch primitives.

$$P(I_l|I_r, V_1, R) \propto \prod_{i=1}^{n_e} L_{\pi_i}(t_i) \exp[-\sum_{r_j \in R} c(r_j)] \tag{7.20}$$

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:

$$E_c(R) = \sum_{r_j \in V_0} (d_j - \mu_j)/2\sigma_j^2$$

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 6×6 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 π_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, $P(V_1) = \prod_{\phi_i\in J} P(\phi_i)$.

7.2.2 The inference algorithm

In our problem formulation, there are two types of variables, discrete and continuous. The discrete variables are

$$\Delta = V_1^d \cup R^d \tag{7.21}$$

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

$$p(V_1, R|I_l, I_r) = p(\Delta, \Gamma, \Psi|I_l, I_r) \tag{7.22}$$

In a MAP formulation, our algorithm needs to perform the following three tasks:

1. Reconstruct the 3D sketch to infer the parameters Γ of the primitives.

2. Label the primitive graph to infer the discrete parameters Δ, 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 Γ, Δ as boundary conditions, to infer Ψ and obtain a dense disparity map D.

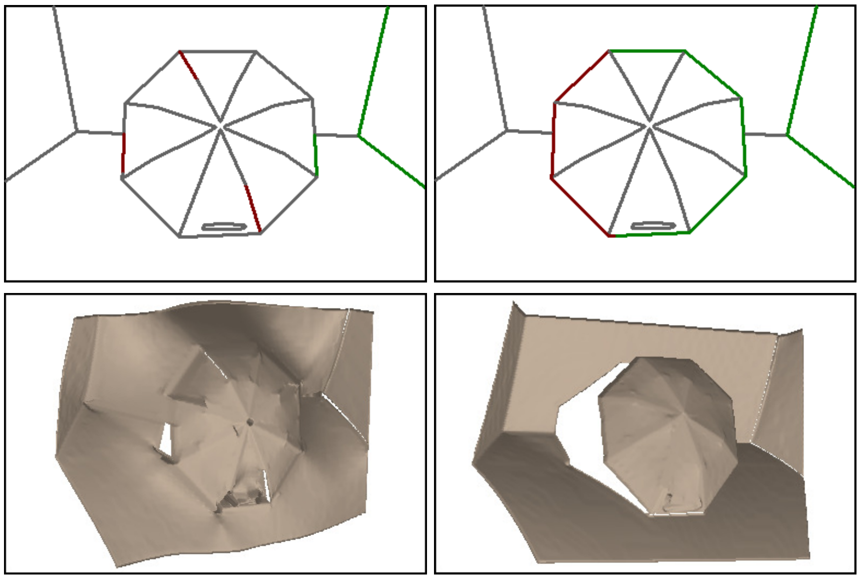


Figure 7.10: Left: An initialization purely based on local information is not satisfactory. Right: By propagating the junction priors along the sketch, a much better initialization can be quickly obtained.

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 ψ_π 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 7.10,left.

A major improvement can be achieved by using the junction prior $P(\phi)$ that has been defined in Section 7.2.1C, 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.

$$P(V_1|I_l, I_r) \propto \prod_{i=1}^{n_e} L_{\pi_i}(t_i) \prod_{\phi_i \in J} P(\phi_i) \tag{7.23}$$

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:

- 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 ϕ 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. (7.23).

The algorithm is run for $10|V_1|$ steps and obtains the initialization result shown in Figure 7.10, 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 Ψ. Observe that in our formulation of the energy, if Δ, Γ are fixed, the conditional $-\log(P(\Psi|\Delta, \Gamma))$ is a quadratic function in all the variables Ψ, so it can be minimized analytically. This implies that Ψ can be regarded as a function on Δ, Γ, $\Psi = \Psi(\Delta, \Gamma)$. This restricts the problem to maximizing the probability $P(\Delta, \Gamma, \Psi(\Delta, \Gamma)|I_l, I_r)$, of much smaller dimensionality.

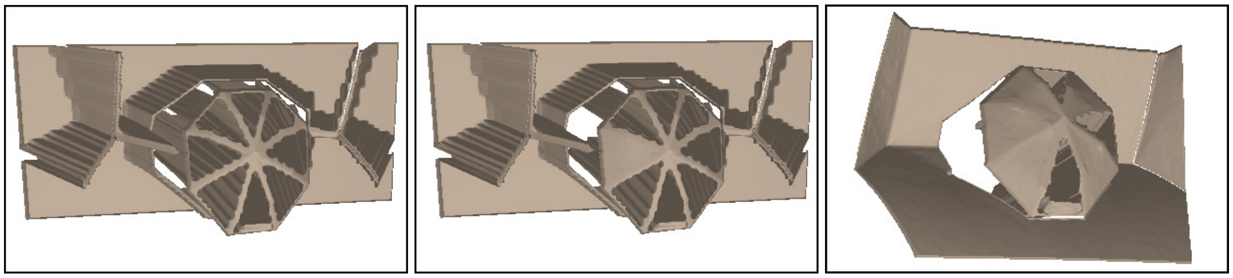


Figure 7.11: The fill-in can be restricted to the connected components bounded by control point boundary primitives. In a few steps, the initial 3D reconstruction before graph labeling is obtained. Shown are the 3D reconstructions after 0,1, and 5 connected components have been updated. The horizontal edges change the disparity at the same time with the interior, because they are not control points.

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 7.11. Additional speedups can be obtained following the approximate thin plate spline methods from [38].

Observe that the update can affect some non-control point edges, such as the horizontal edges from Figure 7.11.

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

$$E(C, l_C) = E_c(C) + \mu_b E_b(C) + \sum_{r \in C \cap R} c(r) \tag{7.24}$$

The full posterior probability can be recovered from the energy of the regions and the junction prior:

$$P(V_1, R|I_l, I_r) \propto \prod_{i=1}^{n_e} L_{\pi_i}(t_i) \exp[-\sum_{C} E(C, l_C)] \prod_{\phi \in J} P(\phi) \tag{7.25}$$

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 Δ and the fill-in variables Ψ.

To maximize $P(\Delta, \Gamma_0, \Psi(\Delta, \Gamma_0)|I_l, I_r)$ we will use a Markov chain Monte Carlo algorithm that will sample $P(\Delta, \Gamma_0, \Psi(\Delta, \Gamma_0)|I_l, I_r)$, and this way obtain the most probable solutions.

At each step, the algorithm proposes, as shown in Figure 7.12, new types for a set of primitives N and junctions J in one move, as follows:

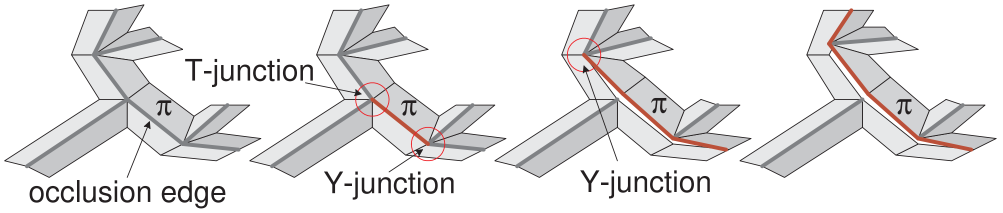


Figure 7.12: Each graph labeling move changes the types of a set of primitives in a consistent manner. First a primitive π is chosen and its type is sampled from the likelihood $L_\pi(t)$, then the adjacent junctions change their type conditional on the chosen type of π, which in turn determine the types 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.

```
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 $L_\pi(t)$.
    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. (7.25).
```

7.2.3 Example results

Experiments are presented in Figure 7.13 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 fifth 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 produce unsatisfactory results. The second and fourth images have free-form surfaces with or without textures from [116]) and [167]. 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 7.11 and the effects of sketch labeling in Figure 7.12.

7.3 2.5D Sketch from Primal Sketch — Shape from Shading

In this section, we briefly present 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

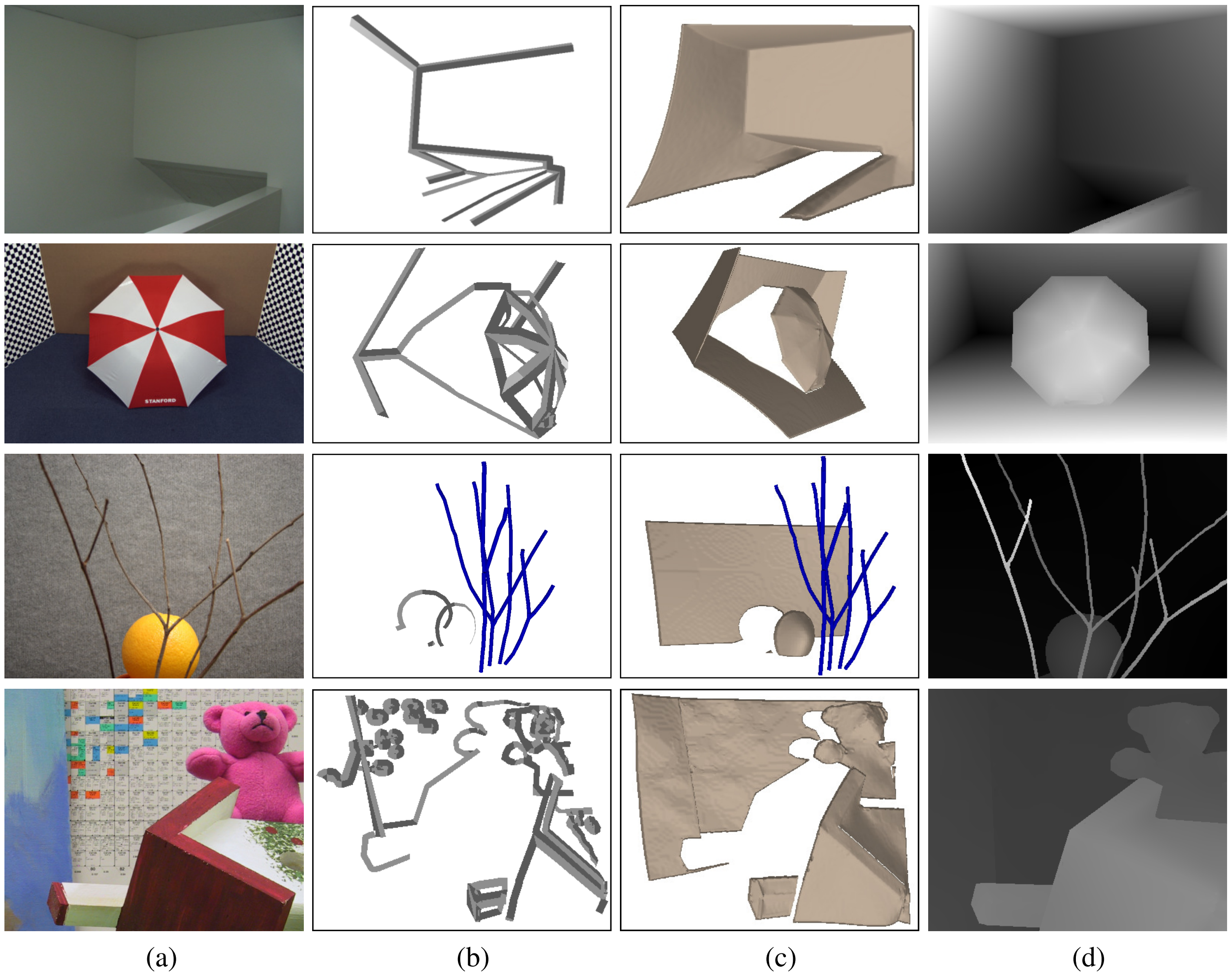


Figure 7.13: Results obtained using our method. (a) left image of the stereo pair, (b) 3D sketch using the primitives, (c) 3D depth map, (d) disparity map.

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 [70, 71]. 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 [128] of computing the $2\frac{1}{2}$D sketch from primal sketch.

7.3.1 Overview of the Two-Layer Generation Model

The dataflow of our method is illustrated in Figure 7.14 and a running example is shown in Figure 7.15. 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 $\mathbf{I}$ on a lattice Λ, we first compute a sketch graph G for the folds by a greedy sketch pursuit algorithm. Figure 7.15.(b) is an exemplary graph G. The graph G

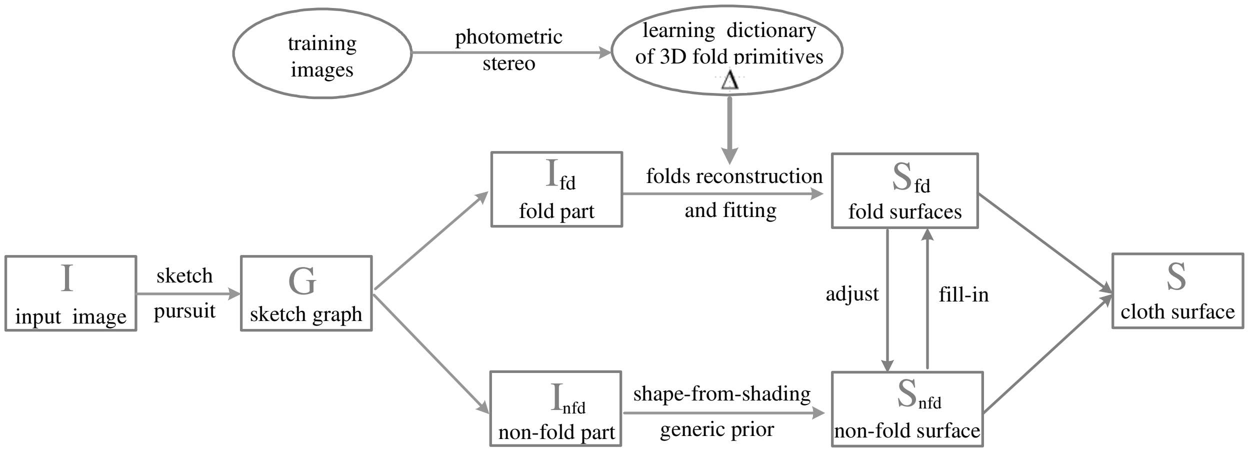


Figure 7.14: The dataflow of our method for computing the 3D surface $\mathbf{S}$ of drapery/cloth from a single image $\mathbf{I}$ using the two-layer generative model. See text for interpretation.

has attributes for the shading and fold primitives. G decomposes the image domain into two disjoint parts: the fold part $\mathbf{I}_{\text{fd}}$ for pixels along the sketch and non-fold part $\mathbf{I}_{\text{nfd}}$ for the remaining flat areas. We estimate the 3D surface $\hat{\mathbf{S}}_{\text{fd}}$ for the fold part by fitting the 3D fold primitives in a fold dictionary Δ_{fd}. Figure 7.15.(c) shows an example of $\mathbf{S}_{\text{fd}}$. This will yield gradient maps $(p_{\text{fd}}, q_{\text{fd}})$ for the fold surface. Then we compute the gradient maps $(p_{\text{nfd}}, q_{\text{nfd}})$ 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 $\mathbf{S} = (\mathbf{S}_{\text{fd}}, \mathbf{S}_{\text{nfd}})$ from the gradient maps (p, q) of both fold part and non-fold part. Therefore the computation of the upper level fold surfaces $\mathbf{S}_{\text{fd}}$ and the lower level flat surface $\mathbf{S}_{\text{nfd}}$ 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 $\mathbf{S}$. 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 quanlitative results.

7.3.2 Results

We test our whole algorithm on a number of images. Figure 7.16 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 [218].

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 $\mathbf{I}_{\text{fd}}$ based on the generative sketch model for the fold areas, the fourth row are the 3D reconstruction results $\mathbf{S}_{\text{fd}}$ for the fold areas, while fifth and sixth rows are the final reconstruction results of the whole cloth surface $\mathbf{S}$ 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.

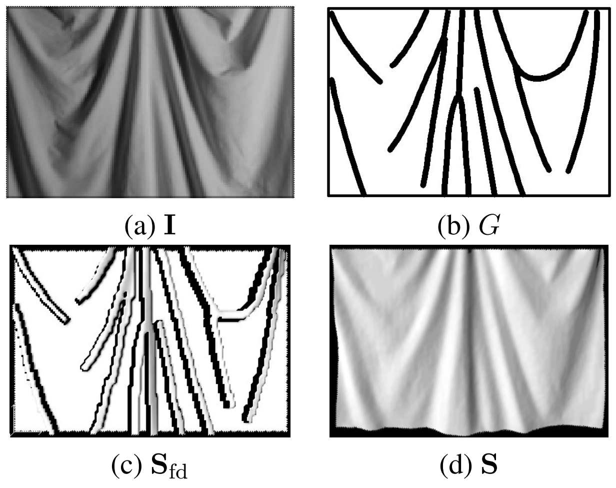

(a) $\mathbf{I}$ (b) G (c) $\mathbf{S}_{\text{fd}}$ (d) $\mathbf{S}$

Figure 7.15: (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.

7.4 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.

7.4.1 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 infrered laser projector combined with a monochrome CMOS sensor, which captures video data in 3D under anyu ambient light conditions. With the help of Kinect, several indoor scene understanding datasets have been introduced. Among them, NYU Depth dataset V2 [170] is the most widely used dataset in recent years. The dataset consists of 1449 RGBD images, gatherd 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 [58] 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 recoding platform is equipped with two high resolution stereo camera systems (grayscale and color), a Velodyne

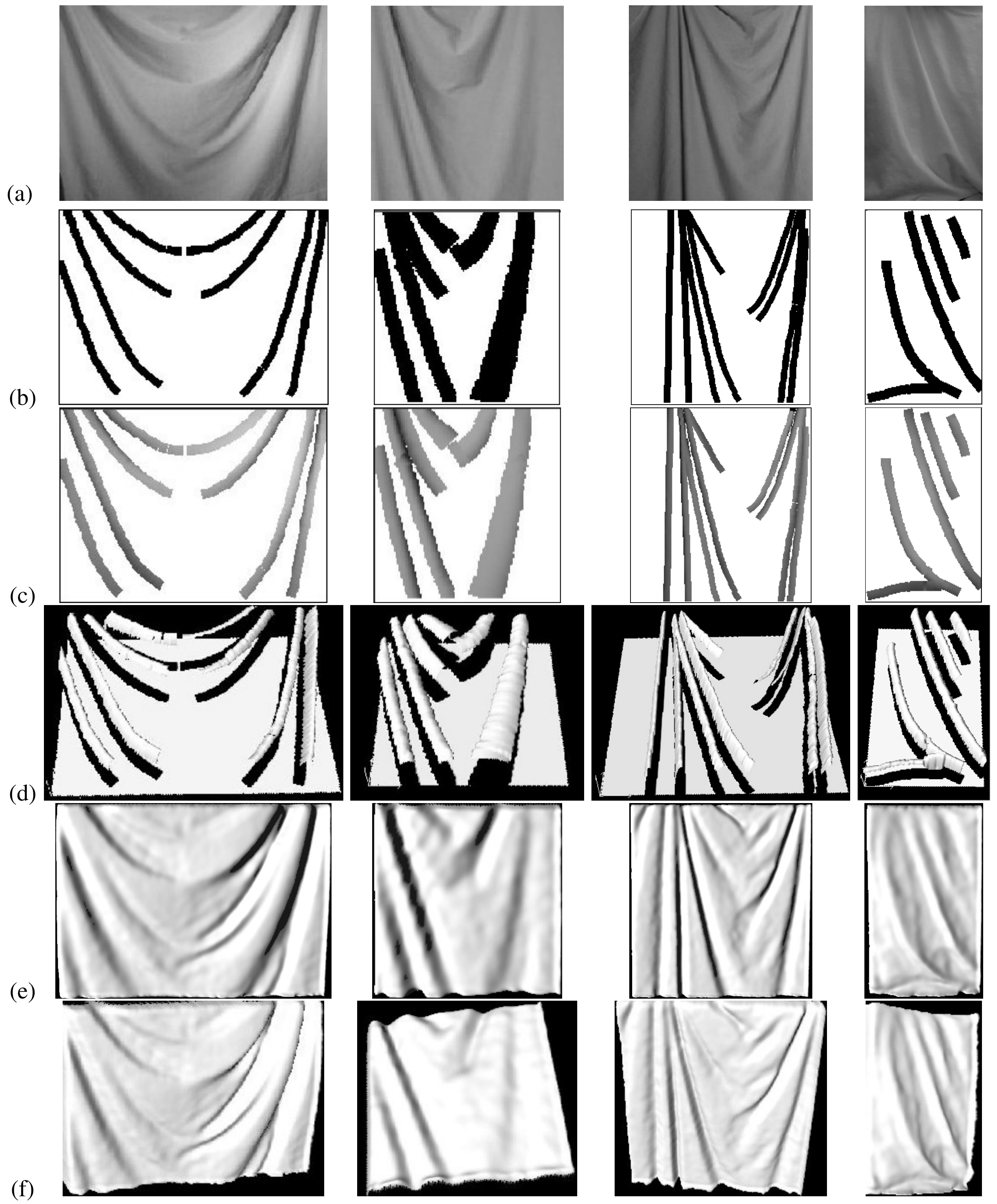


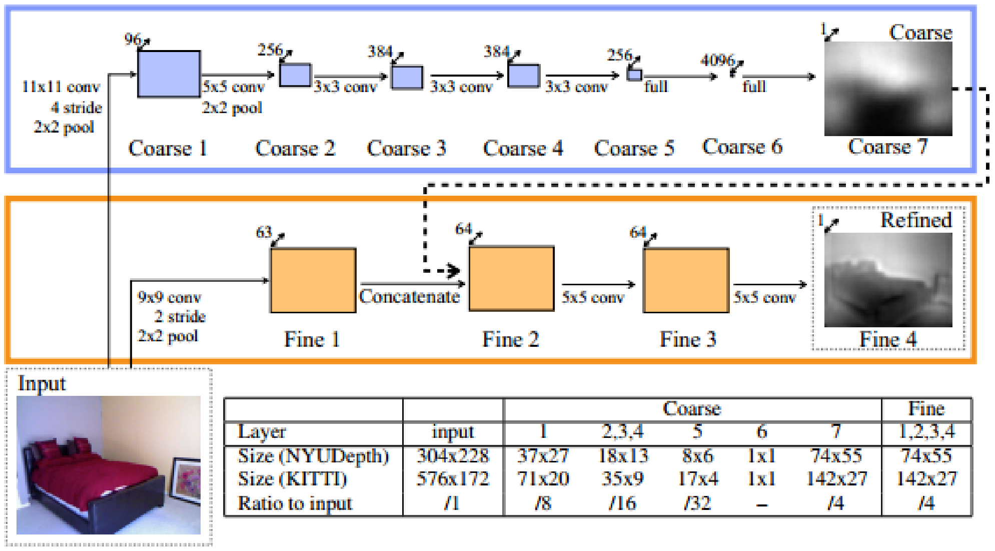
Figure 7.16: (a). Input cloth image. (b). 2d folds and their image domains. (c). Synthesis for 2D fold sketches $\mathbf{I}_{\text{fd}}$. (d). 3D reconstruction $\mathbf{S}_{\text{fd}}$ for fold areas. (e-f). Final reconstructed surface $\mathbf{S}$ in novel views.

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 syustem 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.

7.4.2 Model

In this part, we introduce a representative discriminative approach [41] that uses convolutional neural network to esimate 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 7.17. 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.



		Coarse					Fine
Layer	input	1	2,3,4	5	6	7	1,2,3,4
Size (NYUDepth)	304x228	37x27	18x13	8x6	1x1	74x55	74x55
Size (KITTI)	576x172	71x20	35x9	17x4	1x1	142x27	142x27
Ratio to input	/1	/8	/16	/32	–	/4	/4

Figure 7.17: 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 7.17, 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 7.17. 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 7.17). 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

$$D(y, y^*) = \frac{1}{2n} \sum_{i=1}^{n} (\log y_i - \log y_i^* + \alpha(y, y^*))^2 \tag{7.26}$$

where $\alpha(y, y^*) = \frac{1}{n} \sum_i (\log y_i - \log y_i^*)$ is the value of α that minimizes the error for a given (y, y^*). For any prediction y, e^α 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

$$D(y, y^*) = \frac{1}{n} \sum_i d_i^2 - \frac{1}{n^2} \sum_{i,j} d_i d_j = \frac{1}{n} \sum_i d_i^2 - \frac{1}{n^2} \left(\sum_i d_i \right)^2 \tag{7.27}$$

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

$$L(y, y^*) = \frac{1}{n}\sum_i d_i^2 - \frac{\lambda}{n^2}\left(\sum_i d_i\right)^2 \tag{7.28}$$

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 7.17.
- 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.

7.4.3 Results

We show some quanlitative results in Figure 7.18.

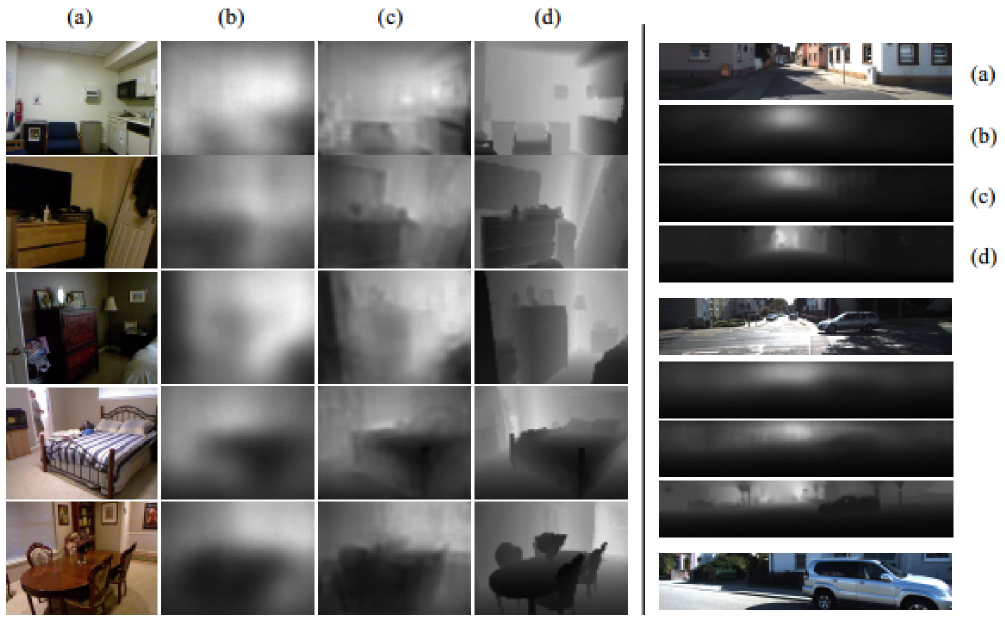


Figure 7.18: Example predictions from our algorithm. NYUDepth [170] on left, KITTI [57] on right. For each image, we show (a) input, (b) output of coarse network, (c) refined output of fine network, (d) ground truth. The fine scale network edits the coarse-scale input to better align with details such as object boundaries and wall edges. Examples are sorted from best (top) to worst (bottom).

8

Learning by Information Projection

In this chapter, we study a general framework for learning a statistical model as an approximation to the true distribution that generates the images.

8.1 Information projection

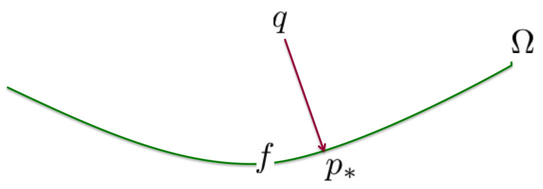


Figure 8.1: In the above illustration, each point is a probability distribution. f is the true distribution that generates the training examples. q is the reference distribution or the null model. The curve Ω consists of all the distributions that reproduce the feature statistics of f. p_* is the projection of q onto Ω.

Suppose we observe training images $\{\mathbf{I}_m, m = 1, ..., M\} \sim f(\mathbf{I})$. We want to find a good approximation to the unknown true distribution f that generates the training examples. Suppose M is large so that we can estimate the expectations with respect to f accurately from the training examples.

Suppose we start from a reference distribution $q(\mathbf{I})$, for instance, the white noise distribution. Suppose have a set of features $H(\mathbf{I}) = (H_k(\mathbf{I}), k = 1, ..., K)$. We can estimate

$$\mathcal{E}_f[H(\mathbf{I})] \approx \frac{1}{M}\sum_{m=1}^{M} H(\mathbf{I}_m). \tag{8.1}$$

$\mathcal{E}_f[H(\mathbf{I})]$ is all that we know about the unknown f as far as the feature H is concerned.

We want to find a distribution p to be an approximation to the unknown distribution f. Such a distribution should better reproduce the feature statistics that we care about, i.e.,

$$\mathcal{E}_p[H(\mathbf{I})] = \mathcal{E}_f[H(\mathbf{I})]. \tag{8.2}$$

Let us call such a p an eligible distribution. Let

$$\Omega = \{p : \mathcal{E}_p[H(\mathbf{I})] = \mathcal{E}_f[H(\mathbf{I})]\} \tag{8.3}$$

be the family of all the eligible distributions. Clearly $f \in \Omega$. See Fig. 8.1 for an illustration, where 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 Ω consists of all the distributions that reproduce the feature statistics of f.

Since we start from q, we want to find a distribution in Ω, so that it has the minimum distance from q. That is, we want to find an eligible distribution that can be obtained from a minimum modification of q, so that we won't introduce artificial features beyond H. Let p_* be such a distribution. We can think of p_* as the projection of q onto the family Ω, and we call it the information projection. More specifically, we want to find

$$p_* = \arg\min_{p \in \Omega} KL(p|q). \tag{8.4}$$

8.1.1 Orthogonality and duality

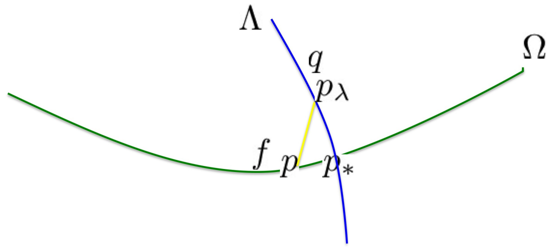


Figure 8.2: The eligible family Ω and the model family Λ are orthogonal, $\Omega \perp \Lambda$, because for any $p \in \Omega$, $p_\lambda \in \Lambda$, $KL(p|p_\lambda) = KL(p|p_*) + KL(p_*|p_\lambda)$, where $p_* = \Omega \cap \Lambda$ is the intersection.

In order to solve for $p_* = \arg\min_{p \in \Omega} KL(p|q)$, we can use Langevin multiplier as we did for the FRAME model. In this chapter, we shall adopt an approach that is less direct but is more geometrically meaningful. We can find a dual minimization problem by introducing another family of distributions that we call model family. Specifically, we define the following exponential family models:

$$p(\mathbf{I}; \lambda) = \frac{1}{Z(\lambda)} \exp\left[\langle \lambda, H(\mathbf{I}) \rangle\right] q(\mathbf{I}), \tag{8.5}$$

where $\lambda = (\lambda_k, k = 1, ..., K)$, $\langle \lambda, H(\mathbf{I}) \rangle = \sum_{k=1}^{K} \lambda_k H_k(\mathbf{I})$, and

$$Z(\lambda) = \int \exp\left[\langle \lambda, H(\mathbf{I}) \rangle\right] q(\mathbf{I}) = \mathcal{E}_q[\exp\left[\langle \lambda, H(\mathbf{I}) \rangle\right]] \tag{8.6}$$

is the normalizing constant. For simplicity, we write $p(\mathbf{I}; \lambda)$ as p_λ. Let

$$\Lambda = \{p_\lambda, \forall \lambda\} \tag{8.7}$$

be the model family. Clearly $q \in \Lambda$ with $\lambda = 0$. See Fig. 8.2 for an illustration, where the model family Λ 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, we shall show that $p_* = \arg\min_{p \in \Omega} KL(p|q)$, which is the projection that we are after. The key is that $\Omega \perp \Lambda$, that is Ω is orthogonal to Λ, in the sense of the following Pythagorean theorem [31]:

Theorem 1. *For any $p_\lambda \in \Lambda$ and any $p \in \Omega$, we have*

$$KL(p|p_\lambda) = KL(p|p_\star) + KL(p_\star|p_\lambda). \tag{8.8}$$

Proof:

$$KL(p|p_\lambda) = \mathcal{E}_p[\log p(\mathbf{I})] - \mathcal{E}_p[\log p(\mathbf{I};\lambda)]. \tag{8.9}$$

$$KL(p|p_\star) = \mathcal{E}_p[\log p(\mathbf{I})] - \mathcal{E}_p[\log p(\mathbf{I};\lambda_\star)]. \tag{8.10}$$

$$KL(p_\star|p_\lambda) = \mathcal{E}_{p_\star}[\log p(\mathbf{I};\lambda_\star]] - \mathcal{E}_{p_\star}[\log p(\mathbf{I};\lambda)]. \tag{8.11}$$

Meanwhile, we have $\mathcal{E}_p[\log p(\mathbf{I};\lambda_\star)] = \mathcal{E}_{p_\star}[\log p(\mathbf{I};\lambda_\star]]$, 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 Ω. Thus the result follows.□

The above result leads to the following duality result:

$$p_* = \arg\min_{p\in\Omega} KL(p|q) = \arg\min_{p_\lambda\in\Lambda} KL(f|p_\lambda). \tag{8.12}$$

Thus, we can find solve $p_* = \arg\min_{p\in\Omega} KL(p|q)$ by finding $p_* = \arg\min_{p_\lambda\in\Lambda} KL(f|p_\lambda)$. Since $KL(f|p_\lambda) = \mathcal{E}_f[\log f(\mathbf{I})] - \mathcal{E}_f[\log p(\mathbf{I};\lambda)]$, we have

$$p_* = \arg\min_{p_\lambda\in\Lambda} KL(f|p_\lambda) = \arg\max_{p_\lambda\in\Lambda} \mathcal{E}_f[\log p(\mathbf{I};\lambda)]. \tag{8.13}$$

$\mathcal{E}_f[\log p(\mathbf{I};\lambda)]$ is actually the log-likelihood in the limit.

8.1.2 Maximum Likelihood Implementation

If we observe $\{\mathbf{I}_m, m = 1, ..., M\} \sim f(\mathbf{I})$, then

$$\mathcal{E}_f[\log p(\mathbf{I};\lambda] \approx \frac{1}{M}\sum_{m=1}^{M} \log p(\mathbf{I}_m;\lambda), \tag{8.14}$$

so λ_* can be approximated by the maximum likelihood estimate $\hat\lambda = \arg\max_\lambda L(\lambda)$, where

$$L(\lambda) = \frac{1}{M}\sum_{m=1}^{M} \log p(\mathbf{I}_m;\lambda) = \frac{1}{M}\sum_{m=1}^{M} \langle\lambda, H(\mathbf{I}_m)\rangle - \log Z(\lambda) \tag{8.15}$$

is the log-likelihood function of the exponential family model (8.5).

It can be shown that

$$\frac{\partial}{\partial\lambda}\log Z(\lambda) = \mathcal{E}_\lambda[H(\mathbf{I})], \tag{8.16}$$

where $\mathcal{E}_\lambda$ denotes the expectation with respect to $p(\mathbf{I};\lambda)$.

$$\frac{\partial^2}{\partial\lambda^2}\log Z(\lambda) = \mathrm{Var}_\lambda[H(\mathbf{I})], \tag{8.17}$$

where Var_λ denotes the variance with respect to $p(\mathbf{I};\lambda)$. Thus

$$\frac{\partial}{\partial\lambda}L(\lambda) = \frac{1}{M}\sum_{m=1}^{M} H(\mathbf{I}_m) - \mathcal{E}_\lambda[H(\mathbf{I})], \tag{8.18}$$

and

$$\frac{\partial^2}{\partial \lambda^2} L(\lambda) = \mathrm{Var}_\lambda[H(\mathbf{I})]. \tag{8.19}$$

That is, $L(\lambda)$ is a concave function with a unique maximum, provided that $\mathrm{Var}[H(\mathbf{I})]$ is positive definite, which is the case if the components of $H(\mathbf{I})$ are linearly independent. At the maximum $\hat{\lambda}$,

$$\mathcal{E}_{\hat{\lambda}}[H(\mathbf{I})] = \frac{1}{M} \sum_{m=1}^{M} H(\mathbf{I}_m), \tag{8.20}$$

where $\mathcal{E}_\lambda$ denotes the expectation with respect to $p(\mathbf{I}; \lambda)$. Thus, at maximum likelihood estimate, the model reproduces the observed feature statistics.

If $M \to \infty$, $L(\lambda) \to \mathcal{E}_f[\log p(\mathbf{I}; \lambda)]$, and at the maximum, $\mathcal{E}_f[H(\mathbf{I})] = \mathcal{E}_{\lambda_*}[H(\mathbf{I})]$, i.e., $p_{\hat{\lambda}} \to p_* = \Lambda \cap \Omega$.

We would like to emphasize that the information projection viewpoint is deeper than the maximum likelihood estimation of the exponential family model (8.5). The former provides a justification for the latter, and the latter is to implement the former.

8.1.3 The Minimax Learning Framework

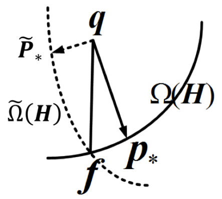


Figure 8.3: The solid curve and the dotted curve illustrate two eligible families defined by two different sets of feature statistics. We should project q onto the solid curve instead of the dotted curve in order to get closer to the target distribution f.

Suppose we have two different sets of features $H(\mathbf{I})$ and $\tilde{H}(\mathbf{I})$, then we have two different eligible families Ω and $\tilde{\Omega}$. If we project the same reference distribution q onto Ω and $\tilde{\Omega}$ respectively, we will get p_* and $\tilde{p}_*$ respectively. In Fig. 8.3, the solid curve illustrates Ω, while the dotted curve illustrates $\tilde{\Omega}$. Due to the Pythagonrean theorem,

$$KL(f|q) = KL(f|p_*) + KL(p_*|q) = KL(f|\tilde{p}_*) + KL(\tilde{p}_*|q). \tag{8.21}$$

Thus, if we want to make $KL(f|p_*)$ small, we need to make $KL(p_*|q)$ big. So if we have many different choices of H, then we should choose the one that maximizes $KL(p_*|q)$. Recall that $p_* = \arg\min_{p\in\Omega} KL(p|q)$, thus we want to solve the following max-min problem:

$$\max_{H} \min_{p\in\Omega(H)} KL(p|q), \tag{8.22}$$

where $\Omega(H)$ 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,

$$\min_{H} \min_{\lambda} KL(f|p_{H,\lambda}), \tag{8.23}$$

where $p_{H,\lambda}$ is the exponential family model defined by H in equation (8.5). Here we make H explicit in $p_{H,\lambda}$ because we are considering different sets of features. Thus, we can maximum the log-likelihood L in (8.15) over both λ and H.

Intuitively, for a given set of features H that defines an eligible family $\Omega(H)$, we want to choose p that is closest to q to avoid adding artificial features that are not in H. Meanwhile, for different sets of features, we should choose the set of features so that the change from q to the corresponding p_* is the biggest.

The minimax entropy learning [224] is a special case of the above learning scheme, where the reference distribution q is the uniform measure.

8.1.4 Model Pursuit Strategies

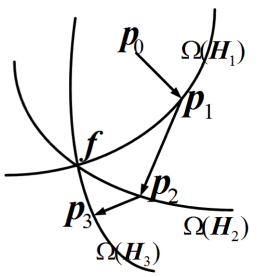


Figure 8.4: Learning a sequence of distributions p_k to approach the target distribution f. Each time, we project the current distribution p_{k-1} onto the eligible family defined by H_k to obtain p_k.

We can select $H(\mathbf{I}) = (H_k(\mathbf{I}), k = 1, ..., K)$ 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. 8.4 illustrates the idea of sequential projection. We start from $p_0 = q$, the reference distribution or the null model. After selecting the first feature H_1, we get the eligible family $\Omega(H_1)$ that consists of all the distributions that reproduce $\mathcal{E}_f[H_1(\mathbf{I})]$. Then we project p_0 onto $\Omega(H_1)$ to get p_1. Then we select the second feature H_2, and project p_1 onto $\Omega(H_2)$ to obtain p_2, and so on. Because of the Pythagonrean theorem, each time we get closer to the target f.

The about sequential projection leads to the following greedy strategy to choose H_k sequentially. At each step, we seek the maximum reduction in the distance from the current model to the target distribution. Specifically, let p_{k-1} be the current model. We choose $H_k = \arg\max KL(p_k|p_{k-1})$, which can be implemented by the maximum likelihood of the following exponential family model:

$$p_k(\mathbf{I}) = \frac{1}{Z_k(\lambda_k)} \exp\left[\lambda_k H_k(\mathbf{I})\right] p_{k-1}(\mathbf{I}), \tag{8.24}$$

where p_{k-1} plays the role of the current reference distribution, and both H_k and λ_k are obtained by maximizing the likelihood function of (8.24) as a function of H_k and λ_k. In the end, we obtain the model of the form (8.5).

In the above discussion, we assume that we have a large dictionary of features $\{H_i, i = 1, ..., N\}$, and we can select a small number of them from this large dictionary. A related strategy for feature selection is via ℓ_1 regularization as in basis pursuit [19] or Lasso [181]. Specifically, we assume the following full model (instead of the final selected model)

$$p(\mathbf{I}; \lambda) = \frac{1}{Z(\lambda)} \exp\left[\sum_{i=1}^{N} \lambda_i H_i(\mathbf{I})\right] q(\mathbf{I}), \tag{8.25}$$

where $\lambda = (\lambda_i, i = 1, ..., N)$ is a long vector. We assume the vector λ is 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, we can perform model selection by maximizing the ℓ_1-regularized log-likelihood, $L(\lambda) + \rho|\lambda|$, where $|\lambda| = \sum_{i=1}^{N} |\lambda_i|$ is the ℓ_1 norm of λ, and ρ is a tuning constant. The maximization of the penalized log-likelihood $L(\lambda) + \rho|\lambda|$ can be accomplished by an epsilon-boosting algorithm [54, 156], where at each step, we choose the component of $L'(\lambda)$ that has the maximum magnitude, and then we update this component by a small amount ϵ.

It is also possible that the dictionary of the features are parametrized by some continuous parameters γ, so the model is

$$p(\mathbf{I}; \theta) = \frac{1}{Z(\theta)} \exp\left[\sum_{i=1}^{N} \lambda_i H_i(\mathbf{I}; \gamma)\right] q(\mathbf{I}). \tag{8.26}$$

where $\theta = (\lambda, \gamma)$. We can learn both λ and γ by maximum likelihood.

8.2 A Unifying View

8.2.1 Relation to Discriminative Learning

Suppose we observe negative examples from the reference distribution $q(\mathbf{I})$, and we observe positive examples from the model $p(\mathbf{I}; \lambda)$ in (8.5). Let α be the prior probability that we observe a positive example. Then the posterior probability that an example $\mathbf{I}$ is a positive example is

$$p(+|\mathbf{I}) = \frac{1}{1 + \exp\left[-\sum_{k=1}^{K} \lambda_k H_k(\mathbf{I}) - b\right]}, \tag{8.27}$$

where $b = \log[\alpha/(1 - \alpha)]$. This is a logistic regression model. If we observe examples from multiple categories, we will have a multinomial logistic regression.

The learning method in the previous section can be considered a generative version of adaboost [48].

8.2.2 Learning FRAME

So far, we have been rather abstract about the features. In this section, we shall illustrate the idea of information projection using concrete examples of learning two-dimensional distributions where the feature statistics are linear projections or filter responses.

Fig. 8.5 illustrates two examples of information projection. The training examples $\{\mathbf{I}_m, m = 1, ..., M\}$ are two-dimensional, i.e., they are images of two pixels. The scatterplot of the data forms a two-dimensional cloud of points. The features are of the form $H_k(\mathbf{I}) = h(\langle \mathbf{I}, B_k \rangle)$, where B_k is also 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, where B_k plays the role of a filter. $h(r)$ is a one-hot indicator vector. Specifically, we divide the range of $\langle \mathbf{I}, B_k \rangle$ into a finite number of L bins, so that $h(r) = (h_l(r), l = 1, ..., L)$. $h_l(r) = 1$ 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 $\{\langle \mathbf{I}_m, B_k \rangle, m = 1, ..., M\}$ projected onto B_k. We can assume the squared length $|B_k|^2 = 1$, and we can discretize the direction B_k in the two-dimensional domain.

We start from the uniform distribution over the two-dimensional domain of $\mathbf{I}$, assumed to be the unit square. Then we apply the model pursuit strategy by selecting B_k, $k = 1, ..., K$. Each time, we select a B_k,

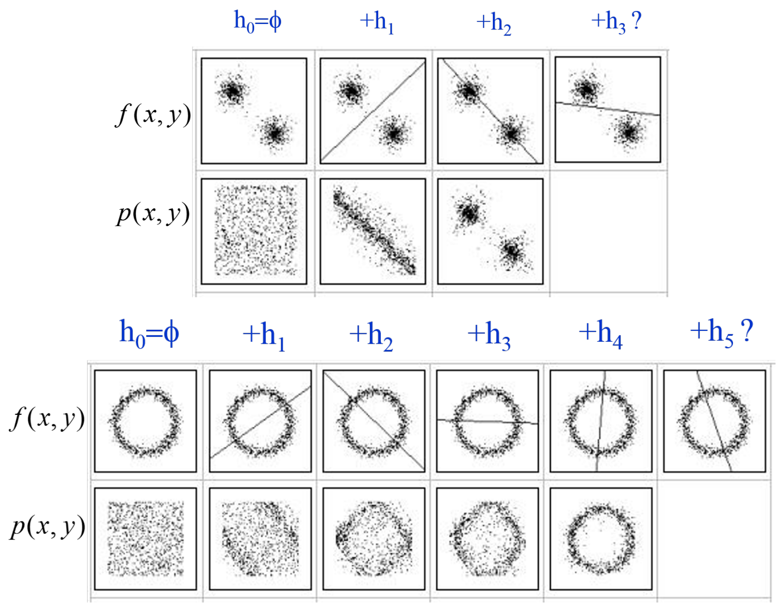


Figure 8.5: Learning two dimensional distributions by information projection. Each step we match the marginal distribution of the data points projected onto a selected vector.

and match the marginal histogram of the projected points. After a number of steps, we pursue a model

$$p(\mathbf{I}; \mathbf{B}, \lambda) = \frac{1}{Z} \exp\left[\sum_{k=1}^{K} \lambda_k h(\langle \mathbf{I}, B_k \rangle)\right] q(\mathbf{I}), \tag{8.28}$$

where $\lambda = (\lambda_k, k = 1, ..., K)$ and $\mathbf{B} = (B_k, k = 1, ..., K)$ is the learned dictionary of projections or filters.

Fig. 8.5 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 [52].

In Fig. 8.5, we start from the uniform distribution. We can also start from Gaussian white noise model with a small variance.

In addition to the pursuit strategy, we may also learn the dictionary $\mathbf{B}$ directly by maximum likelihood, by taking derivatives with respect to both λ and $\mathbf{B}$. In order to take derivative with respect to B_k, we need to make h continuous and differentiable. A possible choice is the rectified linear unit $h(r) = \max(0, r - b)$, where b is the threshold, which can also be estimated by maximum likelihood.

Although the two toy examples are simple, they are very illustrative and they have deep implications. We can extend the model (8.28) to model large images by making the filters B_k convolutional, i.e., B_k is a localized image patch (e.g., 7×7) and we apply B_k around each pixel. If we learn this model from natural images, we will learn Gabor filters and Difference of Gaussian filters as illustrated by Fig. 11.8. Model (8.28) is the simplest FRAME model.

8.2.3 Learning Shape Patterns

In addition to learning image appearance patterns, we can also learn shape patterns by information projection. Fig. 8.6 illustrates an example of learning generic object shapes by adding shape statistics. Fig. 8.7

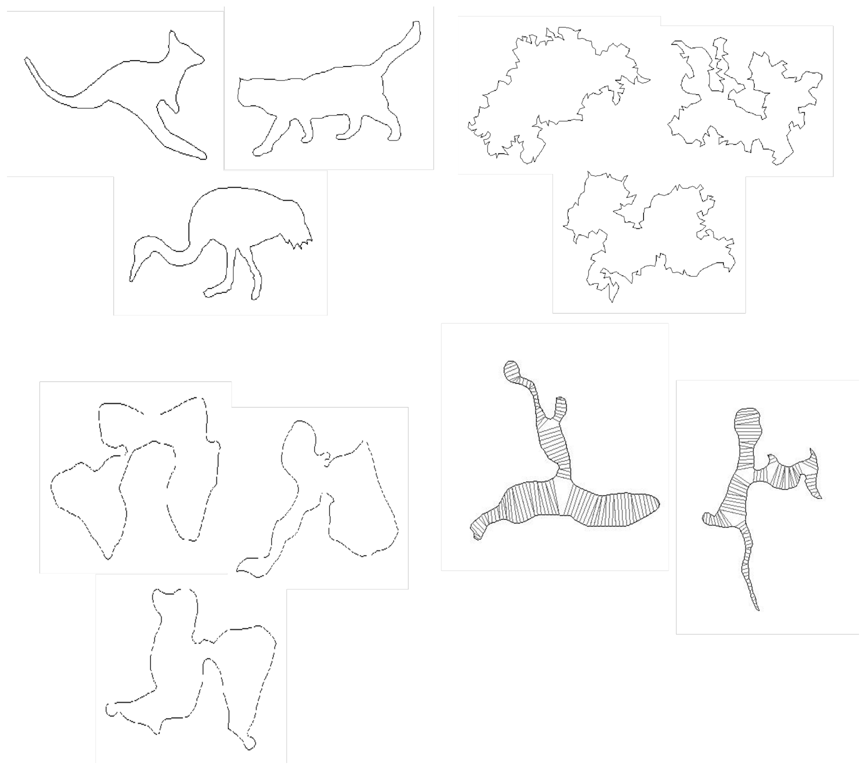


Figure 8.6: Learning a sequence of models for shapes by adding shape statistics.

illustrates an example of learning specific face shapes by adding relevant statistics.

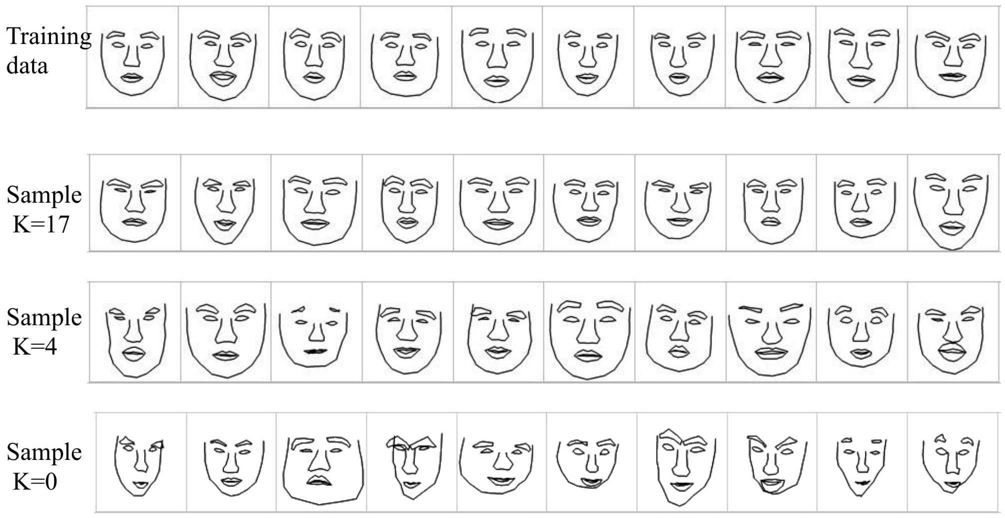


Figure 8.7: Learning a sequence of models for face shapes by adding shape statistics.

9

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. (9.1) 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.

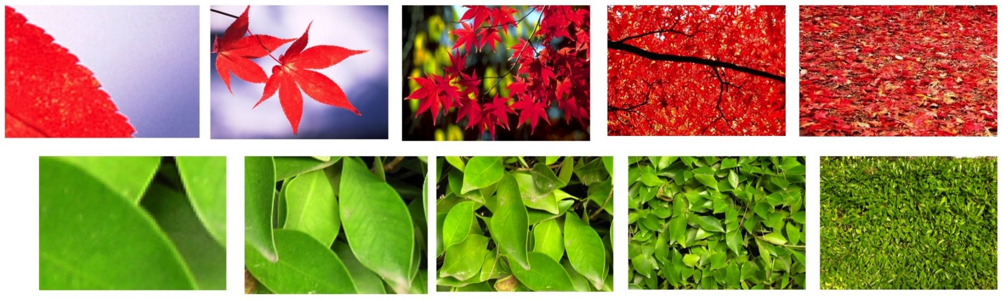


Figure 9.1: Images of the same objects can appear very different at different viewing distance or camera resolution, a phenomenon we call information scaling.

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.

9.1 Image Scaling

To give the reader some concrete ideas, we first study information scaling empirically by experimenting with the so-called dead leaves model.

Model and assumptions

The dead leaves model [130] was used by Lee, et al. [113] in their investigation of image statistics of natural scenes. The model was also previously used to model natural images. For our purpose, we may consider

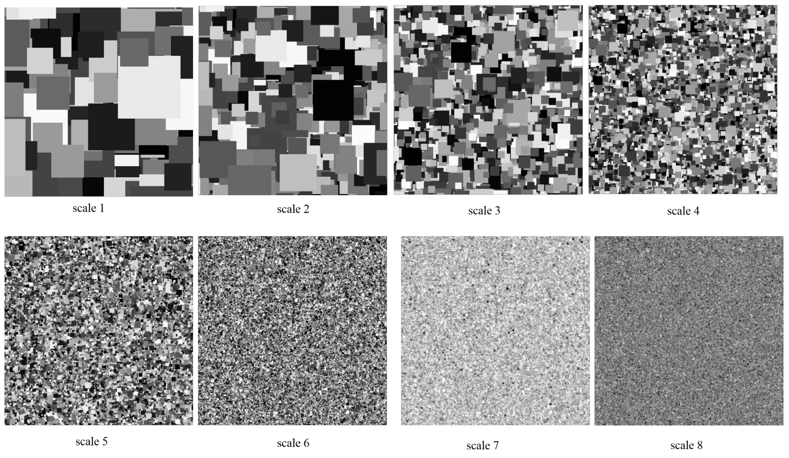


Figure 9.2: Pictures of the simulated ivy wall taken at 8 viewing distances. The viewing distance of the $i+1$-st image is twice that of the i-th image.

that the model describes an ivy wall covered by a large number of leaves of similar sizes. See Fig. (9.2) 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) follow a Poisson process in $\mathbf{R}^2 \times [0, z_{\max}]$. We assume that the intensity of the Poisson process λ 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 $f(r)drda\lambda dxdydz$.

Lee et al. (2001) showed that this Poisson process is scale invariant under the assumption that $[r_{\min}, r_{\max}] \to [0, \infty]$. 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 $[r_{\min}, r_{\max}] \to [0, \infty]$, $[r_{\min}/s, r_{\max}/s] \to [0, \infty]$ 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 Ω 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 lights from a small neighborhood of Ω. As a simple model of the image formation process, we may divide the continuous domain Ω into a rectangular array of squared windows of length or width σd, where σ 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 D. Let $s = d\sigma$ be the scale parameter of the image formation process, then

$$\mathbf{I}_s(i, j) = \frac{1}{s^2} \int_{\Omega_{ij}} W(x, y)dxdy, \ (i, j) \in D, \tag{9.1}$$

which is the average of $W(x, y)$ within window Ω_{ij}. Equation (9.1) can also be written as

$$w_s(x, y) = \frac{1}{s^2} \int W(x', y')g((x - x')/s, (y - y')/s)dx'dy' = W * g_s; \tag{9.2}$$

$$\mathbf{I}_s(i, j) = w_s(u + is, v + js), \tag{9.3}$$

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 (9.2) is smoothing: w_s is a smoothed version of W. Equation (9.3) is subsampling: $\mathbf{I}_s$ is a discrete sampling of w_s. To be more general, g in Equation (9.2) 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 σ. If we increase s by increasing the viewing distance or zooming out the camera, then both the size of the scope Ω and the size of the windows Ω_{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 2×2 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.

Empirical observations on information scaling

Figure (9.2) shows a sequence of 8 images of W taken at 8 viewing distances. The images are generated according to Equation (9.1). 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

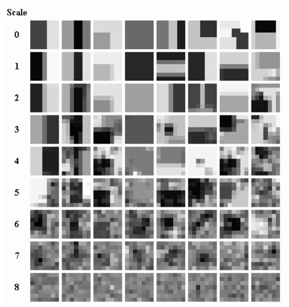


Figure 9.3: The 7×7 local patches taken from the images at different scales.

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.

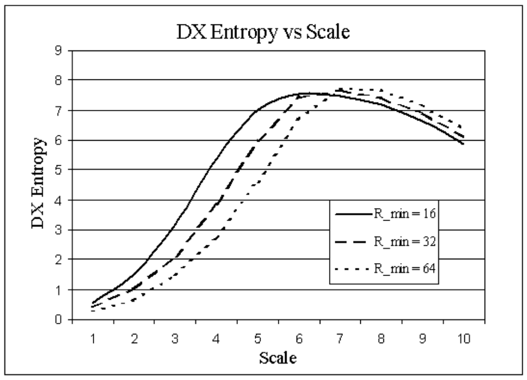
(3) For an image at far distance, the window size of a pixel can be much larger than the average size of the squares, i.e., $s \gg r$. Each pixel covers a large number of leaves, and its intensity value is the average of many leaves. The image is approaching the white noise.

Computer vision algorithms always start from the analysis of local image patches, often at multiple resolutions. We take some local 7×7 image patches from the images at different scales shown in Figure (9.2). 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 random image, there is no much regularity or redundancy in the data, so 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.

The image is compressed by JPEG 2000, and the size of the compressed image file is recorded in terms



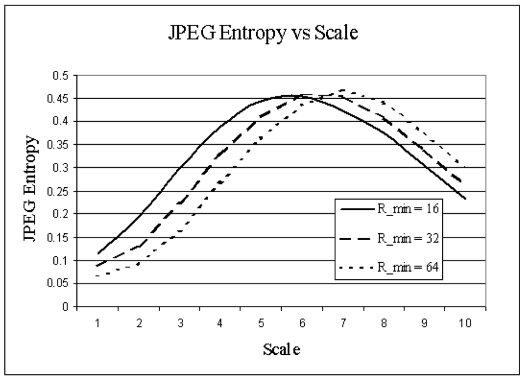


Figure 9.4: The change of statistical properties over scale. (a) JPEG compression rate. (b) Entropy of marginal histogram of $\nabla_x \mathbf{I}$.

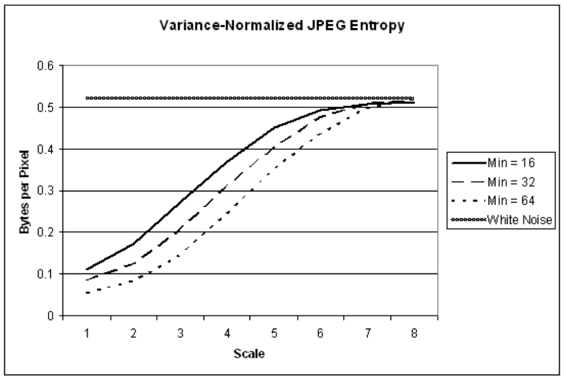
of the number of bits. This number is then divided by the number of pixels to give the compression rate in terms of bits per pixel. Figure (9.4.a) plots this measure in the order of viewing distance for images in Figure (9.2). At near distance, the randomness is small, meaning that the image is quite 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 [**?**]. It is also extensively used in image processing. We make a marginal histogram of $\{\nabla_x \mathbf{I}(i,j), (i,j) \in D\}$ and compute the entropy of the histogram. Figure (9.4.b) plots this entropy over the order of distance for images in Figure (9.2). The plot behaves similarly as the plot of the JPEG 2000 compression rate.

Variance normalization

The local averaging operation in Equation (9.1) reduces the marginal variance of the image intensities. A more appropriate measure of randomness should be the compression rate of variance-normalized image, so that this measure is invariant of linear transformations of image intensities. Specifically, for an image $\mathbf{I}$, let σ^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 (9.2). Figure (9.5.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 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 is decreasing monotonically towards 0,



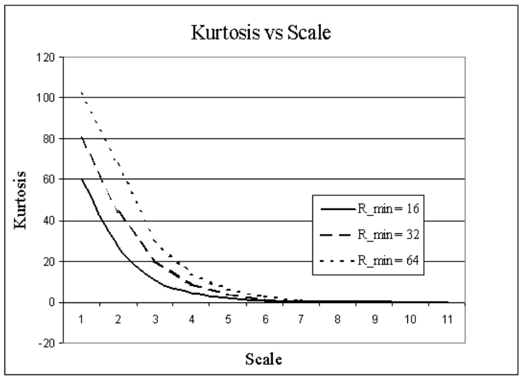


Figure 9.5: (a) The change of JPEG compression rate of the variance-normalized versions of the images in Figure (9.2). (b) The change of kurtosis.

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 [26].

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 σ^2, i.e., $\mathbf{I}(x,y) \sim \text{N}(0,\sigma^2)$ independently. Then

$$\text{KL}(p||q) = -\mathcal{H}(p) - \mathcal{E}_p[\log q(\mathbf{I})] = \mathcal{H}(q) - \mathcal{H}(p) \geq 0. \tag{9.4}$$

The second equation in (9.4) 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 (9.4), 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 (9.4) 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 $\mathbf{I}(x,y), (x,y) \in D$. 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 D_-, 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 (9.2) and (9.3).

(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 D 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 $\min(M,N) \to \infty$,*

$$\bar{\mathcal{H}}(p(\mathbf{J})) - \bar{\mathcal{H}}(p(\mathbf{I})) \to \frac{1}{4\pi^2}\int_0^{2\pi}\int_0^{2\pi}\log|\hat{g}(\omega)|d\omega \leq 0, \tag{9.5}$$

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 $\omega_x \in \{2\pi m/M, m = 0,...,M-1\}$ and $\omega_y \in \{2\pi n/N, n = 0,...,N-1\}$. 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

$$\bar{\mathcal{H}}(p(\mathbf{J})) - \bar{\mathcal{H}}(p(\mathbf{I})) \quad = \quad \frac{1}{|D|}\left[\mathcal{H}(p(\hat{\mathbf{J}})) - \mathcal{H}(p(\hat{\mathbf{I}}))\right]$$

$$
\begin{aligned}
&= \frac{1}{MN}\sum_{\omega}\log|\hat{g}(\omega)| = \frac{1}{4\pi^2}\sum_{\omega}\log|\hat{g}(\omega)|\Delta\omega \\
&\to \frac{1}{4\pi^2}\int_0^{2\pi}\int_0^{2\pi}\log|\hat{g}(\omega)|d\omega,
\end{aligned}
$$

as $\min(M,N)\to\infty$, 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

$$
\begin{aligned}
\hat{g}(\omega) &= \sum_{x,y} g(x,y)\exp\{-i(\omega_x x + \omega_y y)\} \\
&= \mathcal{E}_g\left[\exp\{-i(\omega_x X + \omega_y Y)\}\right],
\end{aligned}
$$

where $(X,Y)\sim g(x,y)$. Then,

$$
\begin{aligned}
|\hat{g}(\omega)|^2 &= |\mathcal{E}_g\left[\exp\{-i(\omega_x X + \omega_y Y)\}\right]|^2 \\
&\le \mathcal{E}_g\left[|\exp\{-i(\omega_x X + \omega_y Y)\}|^2\right] = 1.
\end{aligned}
$$

Thus, $\int\log|\hat{g}(\omega)|d\omega\le 0$. 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) [168].

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 D_-, with $|D_-| = |D|/4$.

Theorem 3. *Subsampling effect: The average entropy rate of* $\mathbf{I}_-^{(u,v)}$ *is no less than the entropy rate of* $\mathbf{J}$*,*

$$
\frac{1}{4}\sum_{u,v}\bar{\mathcal{H}}(p(\mathbf{I}_-^{(u,v)})) - \bar{\mathcal{H}}(p(\mathbf{J})) = \bar{\mathcal{M}}(\mathbf{I}_-^{(u,v)},\forall(u,v))\ge 0, \tag{9.6}
$$

where $\mathcal{M}(\mathbf{I}_-^{(u,v)},\forall(u,v)) = \mathrm{KL}(p(\mathbf{J})||\prod_{u,v}p(\mathbf{I}_-^{(u,v)}))$ *is defined as the mutual information among the four subsampled versions, and* $\bar{\mathcal{M}} = \mathcal{M}/|D|$.

Proof:

$$
\begin{aligned}
\sum_{u,v}\mathcal{H}(p(\mathbf{I}_-^{(u,v)})) - \mathcal{H}(p(\mathbf{J})) &= \mathcal{E}\left[\log\frac{p(\mathbf{J})}{\prod_{u,v}p(\mathbf{I}_-^{(u,v)})}\right] \\
&= \mathrm{KL}(p(\mathbf{J})||\prod_{u,v}p(\mathbf{I}_-^{(u,v)})) \\
&= \mathcal{M}(\mathbf{I}_-^{(u,v)},\forall(u,v))\ge 0,
\end{aligned}
$$

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 (9.5) and (9.6):

$$\left\{\frac{1}{4}\sum_{u,v}\bar{\mathcal{H}}(p(\mathbf{I}_-^{(u,v)})) - \bar{\mathcal{H}}(p(\mathbf{I}))\right\} - \left\{\bar{\mathcal{M}}(\mathbf{I}_-^{(u,v)}) + \frac{1}{4\pi^2}\int \log|\hat{g}(\omega)|d\omega\right\} \to 0. \tag{9.7}$$

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 $\sigma^2 = \mathrm{Var}[\mathbf{I}(x,y)]$ and $\sigma_-^2 = \mathrm{Var}[\mathbf{I}_-(x,y)]$ 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

$$\rho^2 = \frac{\sigma_-^2}{\sigma^2} = \frac{1}{4}\sum_{u,v}\mathrm{corr}(\mathbf{I}(x,y), \mathbf{I}(x+u, y+v)) \le 1,\ (u,v) \in \{(0,0),(0,1),(1,0),(1,1)\},$$

so the smoothing operation reduces the marginal variance. Therefore, we can modify (9.7) into

$$\bar{\mathcal{H}}(p(\mathbf{I}'_-)) - \bar{\mathcal{H}}(p(\mathbf{I}')) \approx \bar{\mathcal{M}}(\mathbf{I}_-^{(u,v)}) - \log\rho + \frac{1}{4\pi^2}\int \log|\hat{g}(\omega)|d\omega, \tag{9.8}$$

where the difference between left-hand side and right-hand size converges to 0 as $|D| \to \infty$. In (9.8), 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 (9.8) 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 (9.8) 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.

9.2 Perceptual Entropy

$$\begin{tikzcd}
\mathrm{I}\_ \arrow[dr, "\textbf{imperceptible}"] & \mathrm{W}\_ \arrow[l, "\textbf{statistical}"'] \\
\mathrm{I} \arrow[u, "\textbf{reduction}"'] & \mathrm{W} \arrow[l, "\textbf{deterministic}"'] \arrow[u, "\textbf{reduction}"']
\end{tikzcd}$$

Figure 9.6: 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 $W = ((x_i, y_i, r_i, a_i), i = 1, ..., N)$ be the leaves in Ω 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 γ represents the transformation defined by Equation (9.1) for a scale parameter s.

For convenience, assume that both W and $\mathbf{I}$ are properly discretized. For any joint distribution $p(W, \mathbf{I})$, the conditional entropy $\mathcal{H}(p(W \mid \mathbf{I}))$ is defined as

$$\mathcal{H}(p(W \mid \mathbf{I})) = -\sum_{W,\mathbf{I}} p(W, \mathbf{I}) \log p(W \mid \mathbf{I}). \tag{9.9}$$

$\mathcal{H}(p(W \mid \mathbf{I}))$ 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 (9.9), $\mathcal{H}(p(W \mid \mathbf{I})) = -\sum_W p(W)(\log p(W) - \log p(\mathbf{I})) = \mathcal{H}(p(W)) - \mathcal{H}(p(\mathbf{I}))$. 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 $\mathcal{H}(p(W \mid \mathbf{I}))$ is below a small threshold. In this regime, the representation $\mathbf{I} = \gamma(W)$ is good for both recognition and coding. For recognition, $\mathcal{H}(p(W \mid \mathbf{I}))$ 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 $\mathcal{H}(p(W \mid \mathbf{I}))$ 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 $\mathcal{H}(p(W \mid \mathbf{I}))$ 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 $\mathcal{H}(p(W \mid \mathbf{I}))$.

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 ρ 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 ρ and R are many to one mappings, we have*

$$(1)\ \ \mathcal{H}(p(W_-|\mathbf{I}_-)) \le \mathcal{H}(p(W|\mathbf{I}_-)).$$

$$(2)\ \ p(\mathbf{I}_-|W_-) = \frac{\sum_{W:\rho(W)=W_-;R(\gamma(W))=\mathbf{I}_-} p(W)}{\sum_{W:\rho(W)=W_-} p(W)}.$$

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 (9.6) 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 $\mathrm{KL}(p(W|\mathbf{I})||p(W|F)) = \mathcal{H}(p(\mathbf{I}|F))$.
(2) If $W \sim p(W)$ and $[\mathbf{I}|W] \sim p(\mathbf{I}|W)$, *then* $\mathrm{KL}(p(W|\mathbf{I})||p(W|F)) = \mathcal{M}(W, \mathbf{I}|F)$, *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 measure 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 $-\mathcal{E}_f\left[\mathcal{E}_{p(w|\mathbf{I})}(\log p(w) + \log p(\mathbf{I}|w))\right]$.

Proposition 5. *The average coding length is $\mathcal{E}_f[\mathcal{H}(p(w|\mathbf{I}))] + \mathrm{KL}(f(\mathbf{I})||p(\mathbf{I})) + \mathcal{H}(f)$, 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.

9.3 A Continuous Spectrum

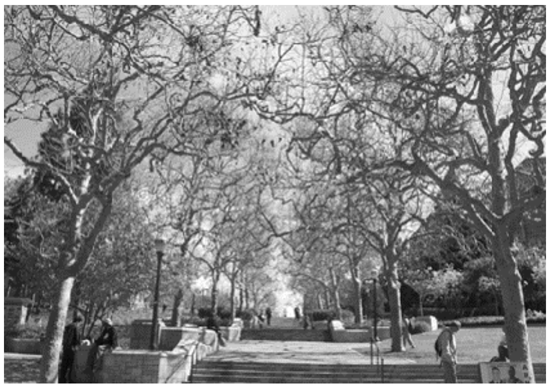


Figure 9.7: The image contains patterns of different complexities, from very simple patterns such as geometric patterns to very random patterns such as leaves at far distance.

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.9.1 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 at 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.

9.4 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 $\mathbf{\Gamma} = \{\Gamma_i, i = 1, ..., N\}$:

$$\mathbf{I} = \sum_{i=1}^{N} c_i \Gamma_i + \epsilon, \quad \Gamma_i \in \mathbf{\Gamma}, \quad C = \{c_i\} \sim p(C), \tag{9.10}$$

where c_i are coefficients, ϵ is the residual, and $\epsilon \sim$ iid $\mathrm{N}(0, \sigma^2)$. 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, θ), i.e., (location, scale, orientation).

For over-complete $\mathbf{\Gamma}$, the coefficients $C = (c_i, i = 1, ..., N)$ 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 ℓ^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 $\{B_k, k = 1, ..., K\} \subset \mathbf{\Gamma}$ 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 Γ_i for some $i \in \{1, ..., N\}$. If we use Gabor bases, then each B_k is a $G_{x,s,\theta}$ for some (x, s, θ). 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 (9.10) 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 $\mathcal{H}(p) = -\int p(x) \log p(x) dx$ [26]. If $p(C)$ is very sparse, then $\mathcal{H}(p(C))$ is small.

Proposition 6. *Let $p(\mathbf{J})$ be the distribution of $\mathbf{J} = \mathbf{\Gamma} C$, $\mathcal{H}(p(\mathbf{J})) \le \mathcal{H}(p(C)) + \log|\det(\mathbf{\Gamma\Gamma}')|/2$.*

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 $\{B_k, k = 1, ..., K\}$. In contract, 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 $\{G_m, m = 1, ..., M\}$ be a set of filters selected to model an image $\mathbf{I}$, where each G_m is a $G_{s,\theta}$ for some (s, θ). 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 $\Delta_1, ..., \Delta_T$, 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, $t = 1, ..., T$. Let $h_{m,t}(\mathbf{I}) = H_{m,t}(\mathbf{I})/|D|$ be the normalized marginal histogram. For simplicity, we write $H_m = (H_{m,t}, t = 1, ..., T)$, and $h_m = (h_{m,t}, t = 1, ..., T)$. Wu, Zhu, and Liu (2000) defined the following image ensemble:

$$\Omega = \{\mathbf{I} : h_m(\mathbf{I}) = h_m(\mathbf{I}_{\text{obs}}), m = 1, ..., M\}. \tag{9.11}$$

If the set of marginal histograms $\{h_m\}$ captures the texture information in $\mathbf{I}_{\text{obs}}$, then all the images in the ensemble Ω 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 Ω. In order to select a set of $\{G_m, m = 1, ..., M\}$ to model a texture image $\mathbf{I}_{\text{obs}}$, we want the resulting Ω defined in (9.11) to have minimum volume $|\Omega|$ or entropy $\log|\Omega|$, or to have maximum log-likelihood $-\log|\Omega|$. $\log|\Omega|$ measures how well $\{G_m, m = 1, ..., M\}$ 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 $|D| \to \infty$, 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

$$p(\mathbf{I}_{D_0}; \Lambda) = \frac{1}{Z(\Lambda)} \exp\{\sum_{1}^{M} \langle \lambda_m, H_m(\mathbf{I}_{D_0}) \rangle\}, \tag{9.12}$$

where $\Lambda = (\lambda_m, m = 1, ..., M)$, $\lambda_m = (\lambda_{m,t}, t = 1, ..., T)$, and $Z(\Lambda)$ is the normalizing constant.

2) According to *asymptotic equipartition property* in information theory, as $|D| \to \infty$, the FRAME model $p(\mathbf{I}_D; \Lambda)$ of the form defined by (10.3) (with D_0 replaced by D) concentrates its probability mass uniformly on the image ensemble Ω of the form (9.11).

In the FRAME model $p(\mathbf{I}; \Lambda)$, Λ 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 $m = 1, ..., M$. 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.*

9.5 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 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

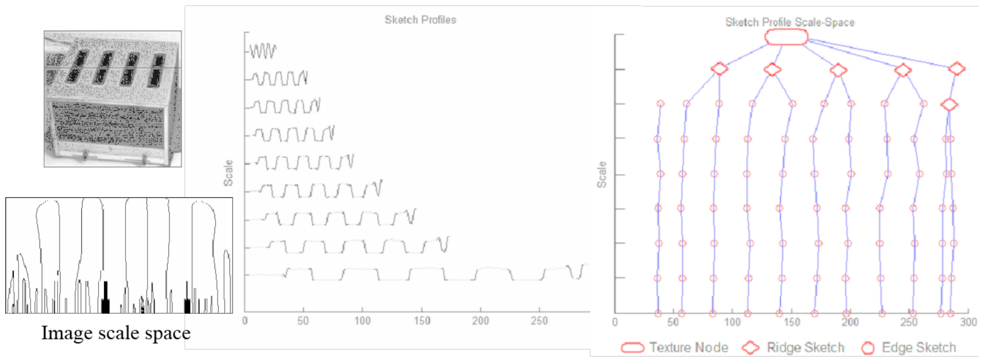


Figure 9.8: Scale-space of a 1D signal. (a) A toaster image from which a line is taken as the 1D signal. (b) Trajectories of zero-crossings of the 2nd derivative of the 1D signal. The finest scale is at the bottom. (c) The 1D signal at different scales. The black segments on the curves correspond to primal sketch primitives (step edge or bar). (d) A symbolic representation of the sketch in scale-space with three types of transitions.

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 we 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.

9.6 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 $U : \Omega_I \to \mathbb{R}$ 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 Ω_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 Ω_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* [77].

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.

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 Ω_I, meaning that the sampling process on f will eventually visit every image $\mathbf{I}$ in the image space

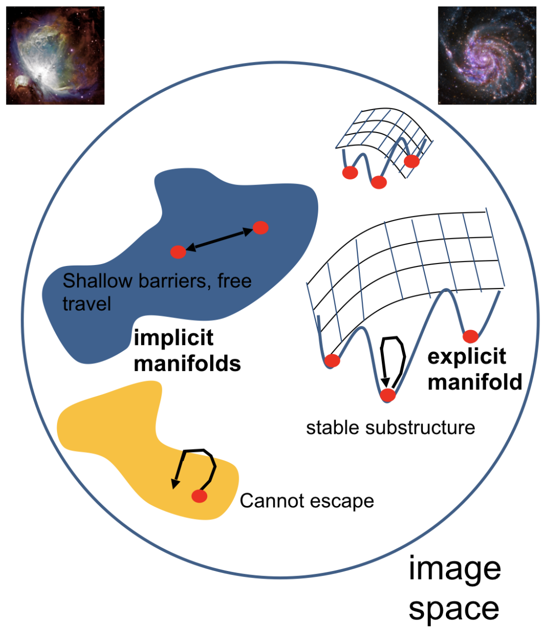


Figure 9.9: The meaningful structures of image density can be intuitively understood as high-density regions in our universe. The majority of the image universe consists of empty space that represents appearances not observed within the training set. Texton-scale images are analogous to star clusters 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.

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 the 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.

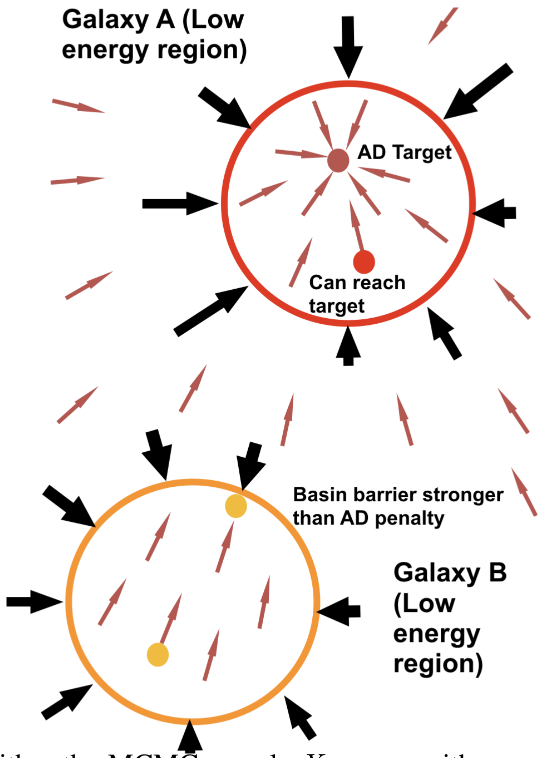
We can intuitively understand the structure of the energy landscape of natural images by considering Ω_I as a "universe" of images. The high energy/low probability regions of U are empty space, which accounts for the vast majority of the volume of Ω_I. The low-energy regions and local modes of U represent high-density regions. Low-energy regions that represent texton-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 9.9.

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 transition 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 transition 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.


Galaxy A (Low energy region)
AD Target
Can reach target
Basin barrier stronger than AD penalty
Galaxy B (Low energy region)

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 α is the strength of magnetization. Sampling is continued until either the MCMC sample X comes with a small Euclidean distance ε 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 α pointing towards x^* for every point $x \neq x^*$. This shows that the gradient of U_{mag} differs from the gradient of U by a magnitude of at most α 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 $\alpha \to 0$, sampling on U_{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 $\alpha \to \infty$ 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 α 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 α is relatively small, while larger values of α 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.

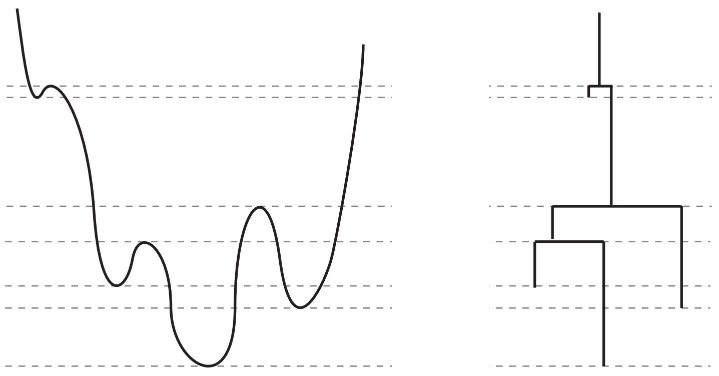


Figure 9.10: Construction of a Disconnectivity Graph (right) from a 1D energy landscape (left). The DG encodes the depth of local minima and the energy at which basins merge. The same procedure can be used to hierarchically cluster image concepts using an energy function U over the image space.

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 to correspond 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 9.10 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 we regularly observe. The macroscopic energy structures of U can effectively distinguish between different

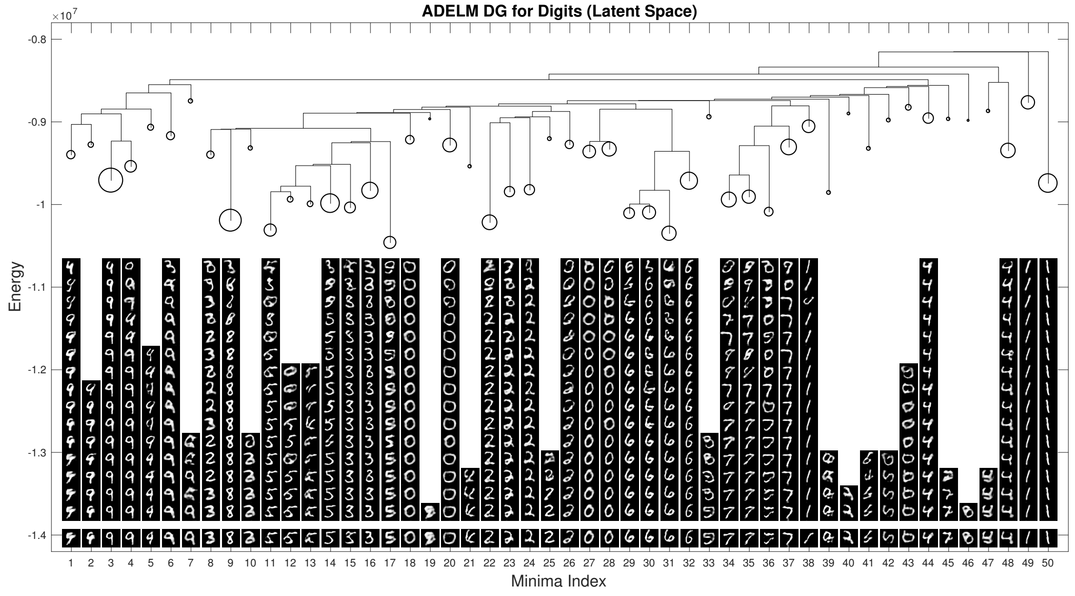


Figure 9.11: Map of the energy landscape of an image density trained with image patches from an ivy texture at four scales. The closest two scales develop recognizable and separate basins in the landscape that represent distinguishable patterns. The furthest two scales do not contain perceptible subgroups and form flat nebula-like basins that encode texture appearance.

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.

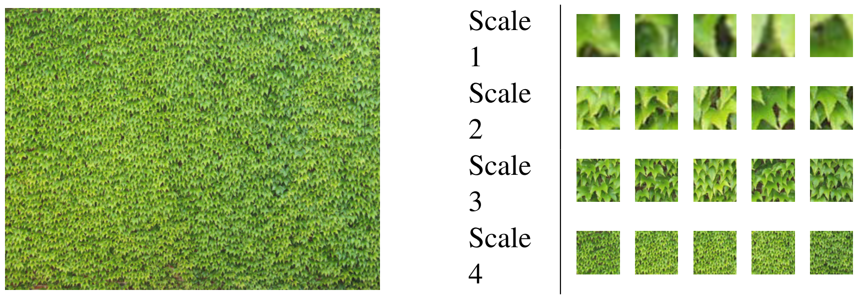


Figure 9.12: Ivy texture image and image patches from four scales. The first two scales contain images which can be clustered by a human. The images cross the perceptibility threshold from texton representation to texture representation between Scale 2 and Scale 3.

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 32×32 pixel image patches of the same ivy texture image from four different scales (see Figure 9.12). 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 9.13). 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.

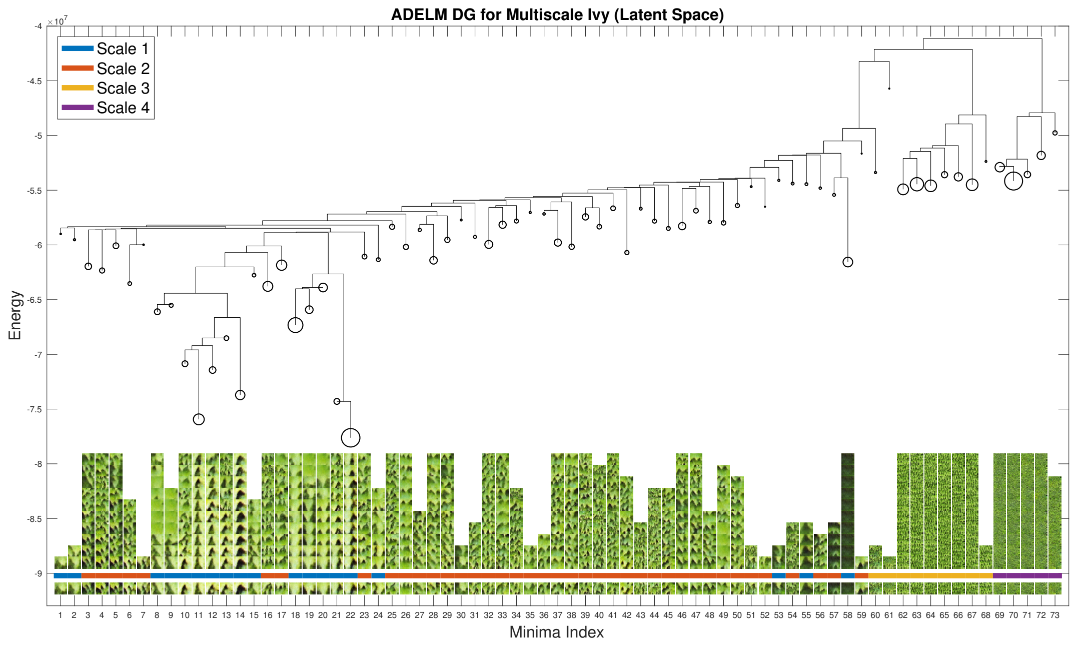


Figure 9.13: Map of the energy landscape of an image density trained with image patches from an ivy texture at four scales. The closest two scales develop recognizable and separate basins in the landscape that represent distinguishable patterns. The furthest two scales do not contain perceptible subgroups and form flat nebula-like basins that encode texture appearance.

10

Image Models with Multilayer Neural Networks

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.

10.1 Deep FRAME

The FRAME (Filters, Random fields, And Maximum Entropy) model 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 X_i via an energy function that is a linear combination of the features,

$$p_\theta(X) = \frac{1}{Z(\theta)} \exp\left[h(X)^{\mathbf{T}}\theta\right] p_0(X), \tag{10.1}$$

where $h(X)$ is the d-dimensional feature vector extracted from X, and θ is the d-dimensional vector of weight parameters. $p_0(X)$ is a known reference distribution such as the white noise model $X \sim \mathrm{N}(0, \sigma^2 I_p)$, or the uniform distribution within a bounded range.

$$Z(\theta) = \int \exp[h(X)^{\mathbf{T}}\theta] p_0(X) dX = \mathcal{E}_{p_0}\{\exp[h(X)^{\mathbf{T}}\theta]\} \tag{10.2}$$

is the normalizing constant ($\mathcal{E}_p$ denotes the expectation with respect to p). It is analytically intractable.

In FRAME model, $h(X)$ consists of histograms of responses from a bank of filters. In a simplified non-convolutional version, $h(X)^{\mathbf{T}}\theta = f(WX) = \sum_{k=1}^{d} f_k(W_k X)$, where W is a $d \times p$ matrix, and W_k is the k-th row of W. WX consists of the d filter responses with each row of W being a linear filter. $(f_k, k = 1, ..., d)$ are d one-dimensional potential functions applied respectively to the d elements

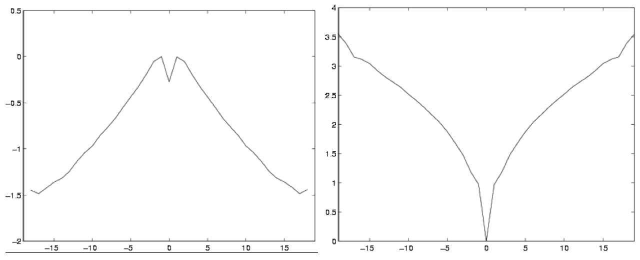


Figure 10.1: Two types of potential functions learned by [221] from natural images. The function on the left encourages big filter responses and creates patterns via reaction, while the function on the right prefers small filter responses and smoothes out the synthesized image via diffusion.

of WX. In the FRAME model, the rows of W are a bank of Gabor wavelets or filters [30]. Given the filters, [221] learned the potential functions $(-f_k, k = 1, ..., d)$ from natural images. There are two types of potential functions as shown in Figure 11.2 taken from [221]. The function on the left encourages big filter responses while the function on the right prefers small filter responses. [221] 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 11.2 is for reaction to create patterns, while the function on the right is for diffusion to smooth out the synthesized image.

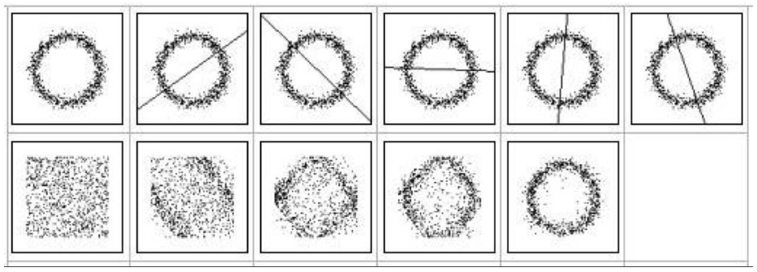


Figure 10.2: Learning a two dimensional FRAME model by sequentially adding rows to W [118]. Each row of W corresponds to a projection of the data. Each step finds the projection that reveals the maximum difference between the observed data and the synthesized data generated by the current model.

The DeepFRAME generalization of the FRAME model is inspired by the successes of deep convolutional neural network [109, 111], and it can be called the deep FRAME model [122]. The filters used in the original FRAME model are linear filters that capture local image features. In the deep FRAME model, 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 a deep convolutional energy-based model [28, 140, 208]. 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. It is possible to learn such a model from natural images [208].

We can generate synthetic images by sampling from the above FRAME models using Markov chain Monte Carlo (MCMC) such as the Langevin dynamics [63, 121], which runs gradient descent on the energy function of the model while adding Gaussian white noises. This sampling scheme was first applied to the original FRAME model by Zhu and Mumford (1998) [223], where the gradient descent part of the dynamics was interpreted as the Gibbs Reaction And Diffusion Equation (GRADE). The Langevin dynamics can be used to sample from deep FRAME model where the gradient can be efficiently computed by back-propagation.

The FRAME model can be written as 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 [84]. 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 the FRAME model follows an analysis by synthesis scheme [67]. We can use the Langevin dynamics to sample from the current model to generate synthetic images. 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.

The sampling of the FRAME model requires iterative MCMC such as Langevin dynamics or Hamiltonian Monte Carlo [138]. A proposed generator model [64, 102, 133, 154] can be recruited as a much more efficient non-iterative sampler that replaces the MCMC sampling by direct ancestral sampling.

The original FRAME model [204, 223, 224] for texture patterns is a stationary or spatially homogeneous Markov random field model [13, 60] defined by the following probability distribution:

$$p(\mathbf{I}; \lambda) = \frac{1}{Z(\lambda)} \exp\left[\sum_{k=1}^{K} \sum_{x \in \mathcal{D}} \lambda_k \left([F_k * \mathbf{I}](x)\right)\right], \tag{10.3}$$

where each $\lambda_k()$ is a one-dimensional non-linear function to be estimated from the training data, $\lambda = (\lambda_k(), k = 1, ..., K)$, and $Z(\lambda)$ is the normalizing constant that makes $p(\mathbf{I}; \lambda)$ integrate to 1. Model (10.3) is stationary because the function $\lambda_k()$ does not depend on position x.

$\lambda_k()$ can be further parametrized, e.g,, $\lambda_k(r) = w_k h(r)$ for some given $h()$, to make (10.3) an exponential family model.

As a Markov random field model or a Gibbs distribution, the FRAME model represents the potential functions in the form of $\lambda_k\left([F_k * \mathbf{I}](x)\right)$, i.e., non-linear transformations of filter responses. The model achieves the maximum entropy among all the distributions with fixed marginal distributions of $F_k * \mathbf{I}(x)$ for $k = 1, ..., K$.

The filters $\{F_k\}$ can be designed, such as the Gabor filters, or be learned from the data together with λ_k.

In the deep FRAME model, the filters are non-linear filters in a ConvNet. We can either use pre-trained ConvNet or train all the filters together with the coefficients. In the following sections, we will first explain why going deep gives us more powerful model from the perspective of matching statistics. Then we will review the ConvNet and finally present the deep FRAME model.

10.1.1 From FRAME to deep FRAME

In this section, we will gradually go from single layer FRAME model trained with pursuit method to multi-layer deep FRAME model trained with gradient desent method. Our discussion will focus on the sufficient statistics used by the model.

As shown in 10.3, the energy term in the FRAME model is the linear combination of $\lambda_k()$, which is a one-dimensional non-linear function of the filter responses. Recall our discussion in Chapter 7, the FRAME model uses the histogram of the filter responses. Assume we denote the kth filter response as I^k:

$$\lambda_k\left(I^k\right) = \sum_{b=1}^{B} \lambda_{kb} H_{kb} \tag{10.4}$$

$$H_{kb} = \delta\left(z_b - I^k\right), \tag{10.5}$$

where $b = 1..B$ represens the B bins we have for the histogram and z_b is the center value of the b_{th} bin. This formulation requires that the fitted probability matches the underlying distribution on the marginal distribution of the filter response. As we have discussed in Chapter 9, this is done by pursuing a sequence of models that get closer and closer to the target distribution. Consider a simple case where data locates in 2D plane (see Figure 10.3). Here, the filter response can be represented as a line in the 2D plane and to do the histogram, we use a group of parallel lines located at different position to partition the filter responses into several bins. The FRAME model requires the fitted distribution matches the underlying distribution in each bin.

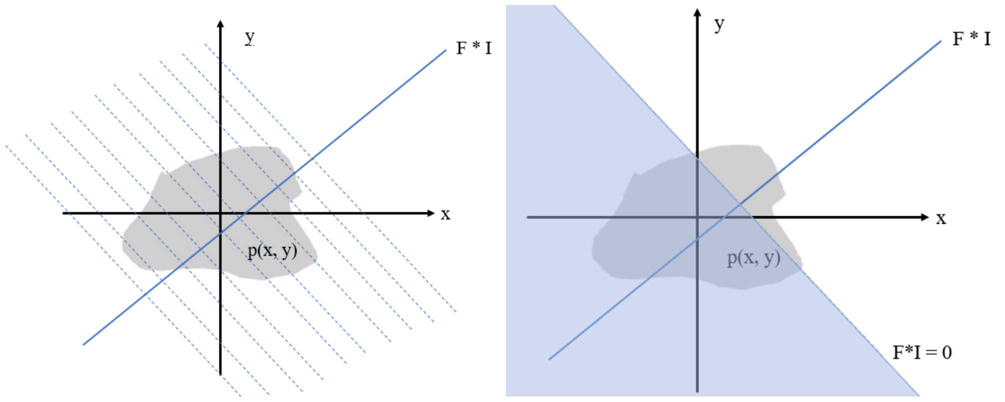


Figure 10.3: In 2D plane, the filter responses can be represented as a straight line. The FRAME model(left) uses histogram and requires the fitted distribution to match the underlying distribution along this line, which is represented by the expectation value in each bin. The ReLU function (right) clips the negative value(shadow area) to zero and requires the fitted distribution to match the underlying distribution on the expectation of this clipped filter response.

Now we consider another $\lambda_k()$ function. Instead of using histogram, we can use ReLu function, which defines as:

$$\lambda_k\left(I^k\right) = ReLU(I^k) = max\left(0, I^k\right), \tag{10.6}$$

In a ReLU function, all the filter responses smaller than zero will be clipped to zero. If we further add a variable bias term to the filter response. Then the values smaller than the negative bias term will be clipped to zero and the ReLU function becomes a line that partitions the space. We can consider the histogram function used in the FRAME model as using multiple lines (represented as different bias terms) as matching statistics after choosing a direction, while the ReLu function only use one line at a time. This enables the model with ReLU function to locate less lines to the directions that are easy to fit.

We can see an example in Figure 10.4. Here we want to fit a distribution consisted of two normal distributions (we have show the same example fitted by FRAME in Chapter 9). The model uses first two filters (represented by two parallel lines partitioned the space at different position) to constrain the distribution into a flat strip area. Then it locates more filters (parallel lines) to further tell apart the two areas. After using 8 filters, the model can almost tell the two distributions and further adding more filters will just do small adjustment. Therefore, we can say that this model uses 2 filters to fit the distribution in one direction and 6 more filters to fit the distribution in another direction. Now we consider what if we use the histogram. In the histogram model, after choosing a direction, we will apply a fixed number of lines to form the bins. Since in the second direction, we has to use 6 lines in total to tell the 2 clusters apart, in total we may need 12 lines to partition the space, which means that we use more statistics to discribe the underlying distribution. In other words, using the ReLU function, because we are applying the line one at a time, we can save some resources on the directions that are easy to discribe.

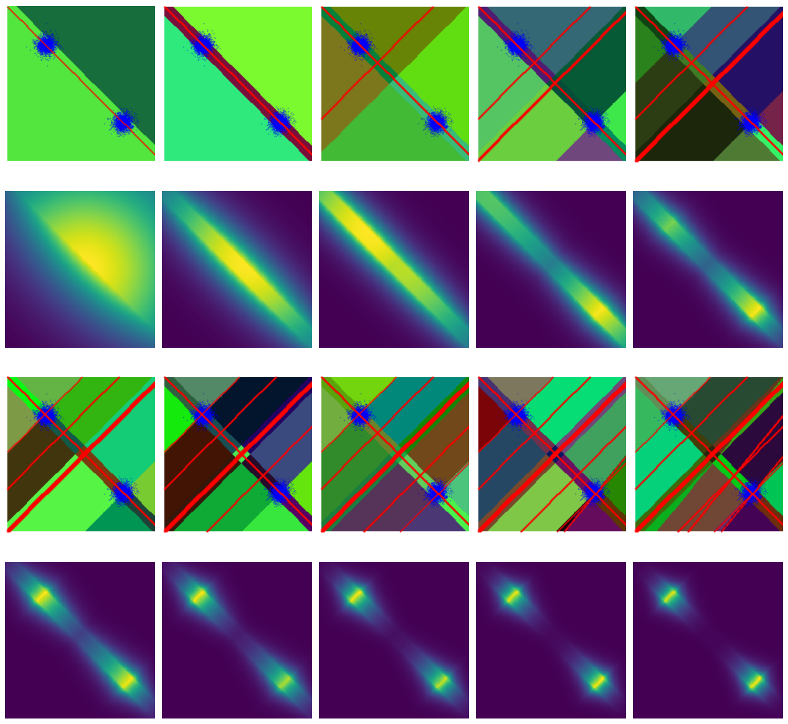


Figure 10.4: Pursuit of one layer FRAME model with ReLU. In the odd lines, from left to right, top to bottom, we show the true data distribution (shown as the points clusters), partition of the space (in the background) and the matching statistics (the lines) by adding filters one by one using pursuit algorithm (each filter correpond to one ReLU function and one line that partitions the space). In the even lines, we show the fitted probability function using the chosen filters.

Now we make another change. Instead of using pursuit algorithm, we change to gradient descent algorithm to find the approperate filters. We fix the number of filters (we can also call them as neurons) and randomly initialize their initial weights. And we use ReLU as our function (also called activation function).

Then we iteratively update the weight of these filters using gradient descent to maximize the probability of the true samples. We fit the same distribution and the result is shown in Figure 10.5. We can see that the models can more or less fit the distribution by partitioning the plane and matching statistics. However, there are two differences here than the pursuit based models. One of them is that the partitions are not at certain directions (i.e. they are not groups of parallel lines). Instead, the partitions seem to have large randomness and some seem to locate at meaningless places. Besides, adding more neurons (i.e. filters) does not always give us better fitted results. We can see that the best result is got at 9 neurons instead of 10 (and you may get totally different results if you run the algorithm again). These observations come from the nature of gradient descent algorithm we use. Unlike in the pursuit-based algorithm where at each time we pick the most unmatched filter response and deal with it, in gradient descent algorithm, we just randomly initialize all the filters and try to find a way to adjust the filter weight together with their coefficients λ_k so that the fitted distribution matches the underlying distribution on these filter responses. There can be the case that some filters just start from or reach some "comfort zone" and stop there. Their responses are automatically matched when we match other filters' responses. Therefore, although they are included in the model, these filters actually do not give us match help in fitting the underlying distribution. The model gets tripped into a local optimum (where all its current filters get matched statitics) and does not try to find the global optimum as the pursuit algorithm does. In this case, the "effective" neurons may not be equal to the number of neurons we use in total.

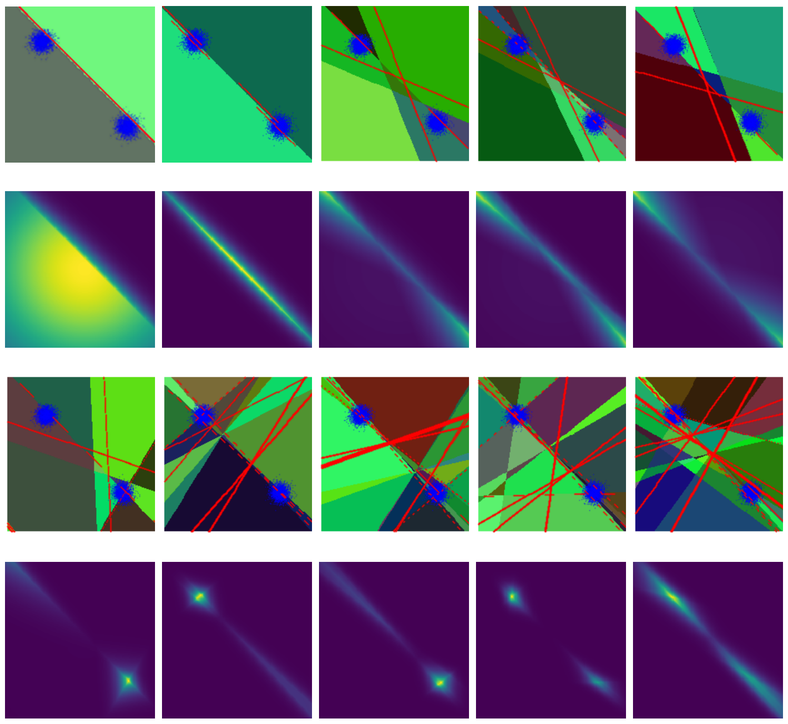


Figure 10.5: Training one layer FRAME model with ReLU using gradient descent. In the odd lines, from left to right, top to bottom, we show the true data distribution (shown as the points clusters), partition of the space (in the background) and the matching statistics (the lines) by adding filters one by one using pursuit algorithm (each filter correspond to one ReLU function and one line that partitions the space). In the even lines, we show the fitted probability function using the chosen filters.

The above discussion seems to support that pursuit algorithm is better than the gradient descent algorithm. However, things may change when we make the model deeper. Here we use multilayer models and want to train the filters at different layers all together. As we discussed above, gradient descent algorithm has its intrinsic randomness, therefore, we only show some typical results here. From the results in Figure 10.6, we can first tell that the fitting result here is pretty good, better than those we get from both pursuit and gradient descent algorithms in one-layer model. More importantly, different from the one-layer cases, now the matching statistics are not limited to just straight lines (see the circle-like matching statistics around each data cluster in the third picture). This is the power of introducing multiple layers. In a multilayer model, the neurons at different layers have different roles. The neurons at lower layers mainly correspond to partition the input space into subspaces. Then in each of these subspace, the neurons at upper layers find their matching statistics. The same statistics at different subspace correspond to lines with different slop and intercept and therefore, they become broken lines. In other words, the existing of lower layer neurons enables the higher layer neurons to choose more flexible shape as matching statistics, which gives the model more fitting power. Note that because of the job-splitting phenomenon, greedily pursuit neurons layer by layer is not suitable for fitting multi-layer model. A neuron that is good at statistics matching may not do a good job in partitioning the space for other neurons. Therefore, the gradient descent algorithm works better in this case.

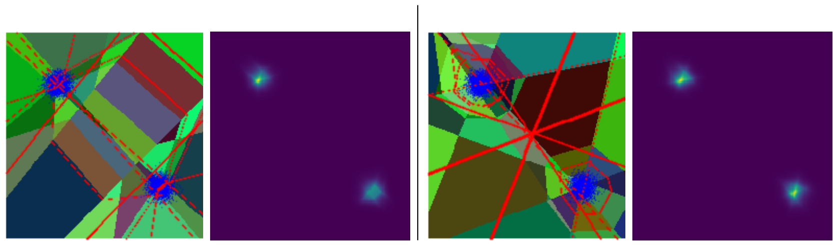


Figure 10.6: Multilayer FRAME model fitted with gradient descent. We use a two-layer model here. The left two pictures come from model that has 4 neurons in the first layer and 5 neurons in the second layer. The right one has 7 neurons in the first layer and 3 neuron in the second layer. For each structure, we show the data distribution, partition of the space, matching statistics and fitted probability function.

The power of multilayer model can be seen more clearly when fitting underlying distributions which have highly irregular shape(texture). Shown in Figure 10.7.The two-layer model can almost capture the shape of underlying distribution when it has 24 neurons in total. However, even using double number of neurons (48 in total), the one-layer model still can not fit the distribution very well. It only captures a triangle-like shape and ignores the subtle shape changes (like the branch in the left side).

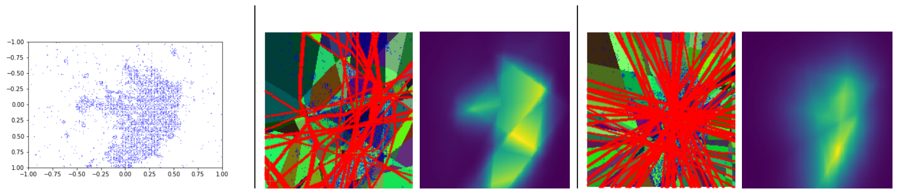


Figure 10.7: Fitting distribution with highly irregular shape. Left: The underlying distribution. Middle: The results of a two-layer model fitted using gradient descent (16 neurons in the first layer ans 8 neurons in the second layer). Right: The results of a one layer model fitted using pursuit algorithm (use 48 neurons in total)

In a summary, we shown in this section that using ReLU function to replace histogram enables the model to choose matching statistics one by one and therefore locates more resource to those hard directions.

Changing pursuit algorithm to gradient descent seems to hurt the performance in single layer model. However, using gradient descent, we can stack multiple layers to get more powerful model, where the lower layer neurons helps the high layer neurons to find more flexible statistics. These observations drive us to push the original FRAME model into deep structure where ConvNet is used.

10.1.2 ConvNet filters

The convolutional neural network (CNN or ConvNet) [111] 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 10.8 for an illustration.

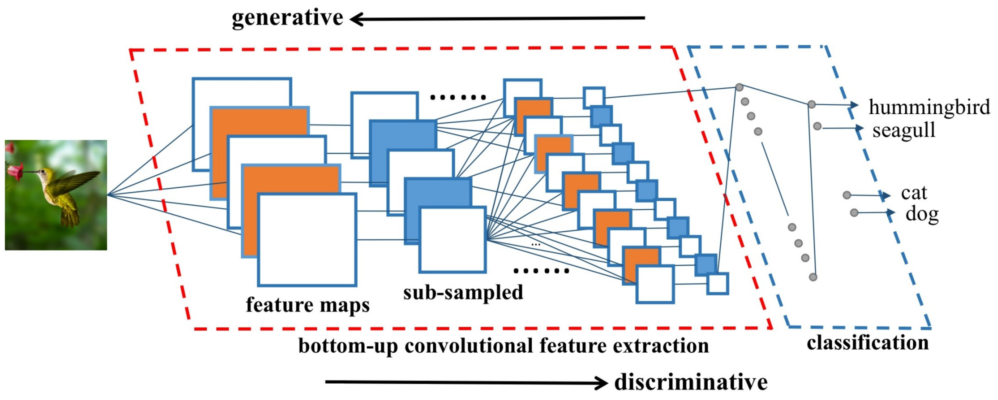


Figure 10.8: Convolutional neural networks consist of multiple layers of filtering and sub-sampling operations for bottom-up feature extraction, resulting in multiple layers of feature maps and their sub-sampled versions. The top layer features are used for classification via multinomial logistic regression. The discriminative direction is from image to category, whereas the generative direction is from category to image. This illustration is adapted from [111].

A ConvNet consists of multiple layers of linear filtering and point-wise non-linear transformation, as expressed by the following recursive formula:

$$[F_j^{(l)} * \mathbf{I}](y) = h\left(\sum_{k=1}^{N_{l-1}} \sum_{x \in \S_l} w_{k,x}^{(l,j)} [F_k^{(l-1)} * \mathbf{I}](y+x) + b_{l,j}\right), \tag{10.7}$$

or

$$\mathbf{I}_j^{(l)}(y) = h\left(\sum_{k=1}^{N_{l-1}} \sum_{x \in \S_l} w_{k,x}^{(l,j)} \mathbf{I}_k^{(l-1)}(y+x) + b_{l,j}\right), \tag{10.8}$$

where $l = 1, ..., L$ indexes the layer, and $\mathbf{I}_j^{(l)} = F_j^{(l)} * \mathbf{I}$ are filtered images or feature maps at layer l. In Figure 10.8, 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.

$\{F_j^{(l)}, j = 1, ..., N_l\}$ are the filters at layer l, and $\{F_k^{(l-1)}, k = 1, ..., N_{l-1}\}$ 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 x in $\sum_x$ is within a local support $\S_l$ (such as a 7×7 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 $h()$. The mostly commonly used non-linear transformation in the modern ConvNets is the rectified linear unit (ReLU), $h(r) = \max(0, r)$ [109]. $\{F_j^{(l)}\}$ are non-linear filters because we incorporate $h()$ 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 $\mathbf{I}^{(l)} = (\mathbf{I}_j^{(l)}, j = 1, ..., N_l)$, which consists of a total of N_l feature maps at layer l, and $j = 1, ..., N_l$. Sometimes, people call $\mathbf{I}^{(l)}$ as a whole feature map 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 10.8 for an illustration of sub-sampling. After a number of layers with sub-sampling, the filtered images or feature maps are reduced to 1×1 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.

10.1.3 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 $\{F_k^{(l)}, k = 1, ..., K\}$ at a certain convolutional layer l of a pre-learned ConvNet, as recursively defined by (10.7). 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 x (again x 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 $h()$. Then the non-stationary deep FRAME model becomes

$$p(\mathbf{I}; w) = \frac{1}{Z(w)} \exp\left[\sum_{k=1}^{K} \sum_{x\in\mathcal{D}} w_{k,x}[F_k^{(l)} * \mathbf{I}](x)\right] q(\mathbf{I}), \tag{10.9}$$

where $q(\mathbf{I})$ is again the Gaussian white noise model (4.24) , and $w = (w_{k,x}, \forall k, x)$ are the unknown parameters to be learned from the training data. Model (10.9) shares the same form as model (4.23) with linear filters, except that the rectification function $h(r) = \max(0, r)$ in model (4.23) is already absorbed in the ConvNet filters $\{F_k^{(l)}\}$ in model (10.9). We can also make model (10.9) stationary by letting $w_{k,x} = w_k$ for all x.

10.1.4 Learning and sampling

The basic learning algorithm estimates the unknown parameters w from a set of aligned training images $\{\mathbf{I}_i, i = 1, ..., n\}$ 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 [214]:

$$w_{k,x}^{(t+1)} = w_{k,x}^{(t)} + \gamma_t \left[\frac{1}{n}\sum_{i=1}^{n}[F_k^{(l)} * \mathbf{I}_i](x) - \frac{1}{\tilde{n}}\sum_{i=1}^{\tilde{n}}[F_k^{(l)} * \tilde{\mathbf{I}}_i](x)\right] \tag{10.10}$$

for every $k \in \{1, ..., K\}$ and $x \in \mathcal{D}$, where γ_t is the learning rate, and $\{\tilde{\mathbf{I}}_i, i = 1, ..., \tilde{n}\}$ 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 $p(\mathbf{I}; w)$, we adopt the Langevin dynamics [63, 121]. Writing the energy function

$$U(\mathbf{I}, w) = -\sum_{k=1}^{K} \sum_{x \in \mathcal{D}} w_{k,x} [F_k^{(l)} * \mathbf{I}](x) + \frac{1}{2\sigma^2} ||\mathbf{I}||^2, \tag{10.11}$$

the Langevin dynamics iterates

$$\mathbf{I}_{\tau+1} = \mathbf{I}_\tau - \delta U'(\mathbf{I}_\tau, w) + \sqrt{2\delta}\epsilon_\tau, \tag{10.12}$$

where $U'(\mathbf{I}, w) = \partial U(\mathbf{I}, w)/\partial \mathbf{I}$. This gradient can be computed by back-propagation. In (11.34), δ is a small step-size, and $\epsilon_\tau \sim \mathrm{N}(0, I_D)$, independently across τ, where I_D is the identity matrix of dimension $D = |\mathcal{D}|$, i.e., the dimensionality of $\mathbf{I}$. ϵ_τ is a Gaussian white noise image whose pixel values follow $\mathrm{N}(0, 1)$ independently. Here we use τ 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 (11.34) is an inner loop within the learning process (11.35). 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 [223], 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.

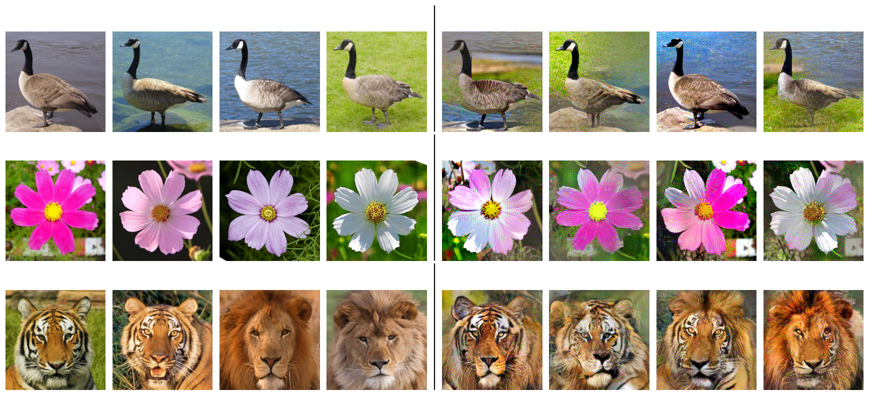


Figure 10.9: Generating object patterns. In each row, the left half displays 4 of the training images (224 $\times$ 224), and the right half displays 4 of the synthetic images. In the last row, the learned model generates hybrid patterns of lion and tiger.

We first learn a non-stationary FRAME model (10.9) 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 10.9 shows some experiments using filters from the 3rd convolutional layer of the VGG ConvNet [173], a commonly used pre-learned ConvNet trained on Imagenet ILSVRC2012 dataset [33]. 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.

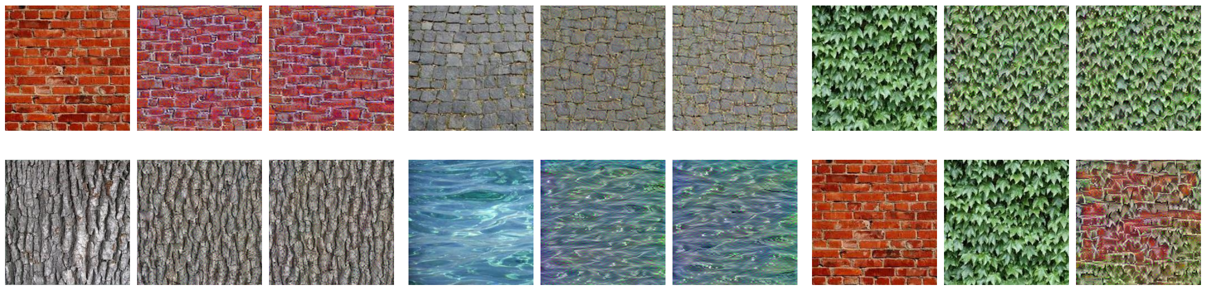


Figure 10.10: Generating texture patterns. For each category, the first image (224 $\times$ 224) is the training image, and the next 2 images are generated images, except for the last 3 images, where the first 2 are the training images, and the last one is the generated image that mixes brick wall and ivy.

Figure 10.10 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.

10.1.5 Learning a new layer of filters

On top of the existing pre-learned convolutional layer of filters $\{F_k^{(l)}, k = 1, ..., K\}$, we can build another layer of filters $\{F_j^{(l+1)}, j = 1, ..., J\}$, according to the recursive formula (10.7), so that

$$[F_j^{(l+1)} * \mathbf{I}](y) = h\left(\sum_{k,x} w_{k,x}^{(j)} [F_k^{(l)} * \mathbf{I}](y + x) + b_j\right), \tag{10.13}$$

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 (10.9) 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 [28], 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 $-\log Z(w)$.

Model (10.9) 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 (10.9) to a convolutional version with a whole new layer of multiple filters

$$p(\mathbf{I}; w) = \frac{1}{Z(w)} \exp\left[\sum_{j=1}^{J} \sum_{x \in \mathcal{D}} [F_j^{(l+1)} * \mathbf{I}](x)\right] q(\mathbf{I}), \tag{10.14}$$

where $\{F_j^{(l+1)}\}$ are defined by (10.13). This model is a product of experts model [78, 158], 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 x. The stationary model for textures is a special case of this model.

Suppose we observe images of non-aligned objects from multiple categories $\{\mathbf{I}_i, i = 1, ..., n\}$, and we want to learn a new layer of filters $\{F_j^{(l+1)}, j = 1, ..., J\}$ by fitting the model (10.14) with (10.13) 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 $L(w) = \frac{1}{n}\sum_{i=1}^{n} \log p(\mathbf{I}_i; w)$ be the log-likelihood where $p(\mathbf{I}; w)$ is defined by (10.14) and (10.13). Then the gradient ascent learning algorithm is based on

$$\frac{\partial L(w)}{\partial w_{k,x}^{(j)}} = \frac{1}{n}\sum_{i=1}^{n}\sum_{y\in\mathcal{D}} s_{j,y}(\mathbf{I}_i)[F_k^{(l)} * \mathbf{I}_i](y+x) - \mathcal{E}_w\left[\sum_{y\in\mathcal{D}} s_{j,y}(\mathbf{I})[F_k^{(l)} * \mathbf{I}](y+x)\right], \quad (10.15)$$

where

$$s_{j,y}(\mathbf{I}) = h'\left(\sum_{k,x} w_{k,x}^{(j)}[F_k^{(l)} * \mathbf{I}](y+x) + b_j\right) \quad (10.16)$$

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 (10.15) admits an EM [32] 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 (11.35), 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.

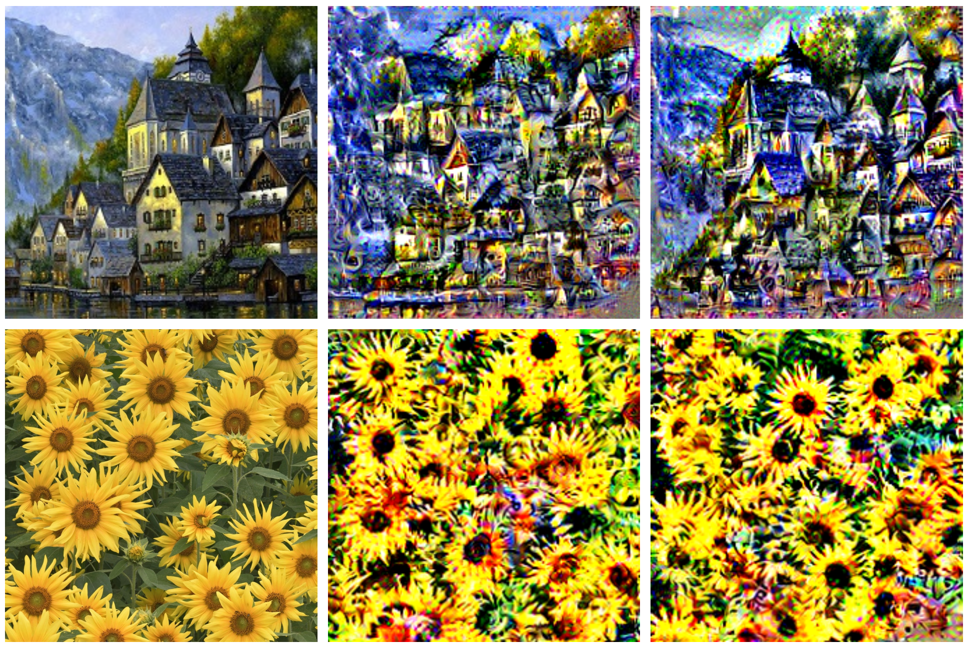


Figure 10.11: Learning a new layer of filters without requiring object bounding boxes or image alignment. For each experiment, the first image (224 $\times$ 224) is the training image, and the next 2 images are generated by the learned model.

Figure 10.11 displays two experiments. In each experiment, the first image (224×224) 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.

10.1.6 Deep convolutional energy-based model

Instead of relying on the pre-trained filters from an existing ConvNet, we can also learn the filters $\{F_k^{(l)}, k = 1, ..., K\}$ themselves. The resulting model is a deep convolutional energy-based model [28, 140, 208],

$$p(\mathbf{I}; w) = \frac{1}{Z(w)} \exp[f(\mathbf{I}; w)] q(\mathbf{I}), \tag{10.17}$$

where $f(\mathbf{I}; w)$ is defined by a ConvNet. In model model (10.14) with (10.13), we have

$$f(\mathbf{I}; w) = \sum_{j=1}^{J} \sum_{x \in \mathcal{D}} [F_j^{(l+1)} * \mathbf{I}](x). \tag{10.18}$$

Using more compact notation, we can define $f(\mathbf{I}; w)$ recursively by

$$\mathbf{I}^{(l)} = h(w_l \mathbf{I}^{(l-1)} + b_l), \tag{10.19}$$

for $l = 1, ..., L$, where $h()$ is applied element-wise. $\mathbf{I}^{(0)} = \mathbf{I}$, and $f(\mathbf{I}; w) = \mathbf{I}^{(L)}$. $\mathbf{I}^{(l)}$ 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 1×1 feature map). $w = (w_l, b_l, l = 1, ..., L)$. We can compare the compact equation (10.19) with the more detailed equation (10.8).

For piecewise linear $h()$, such as $h(r) = \max(0, r)$, the function $f(\mathbf{I}; w)$ is piecewise linear [134, 147]. 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

$$\mathbf{I}^{(l)} = s_l(\mathbf{I}; w)(w_l \mathbf{I}^{(l-1)} + b_l), \tag{10.20}$$

where

$$s_l(\mathbf{I}; w) = \text{diag}(1(w_l \mathbf{I}^{(l-1)} + b_l > 0)), \tag{10.21}$$

i.e., a diagonal matrix of binary indicators (the indicator function is applied element-wise) [147]. Let $s = (s_l, l = 1, ..., L)$ consists of indicators at all the layers, then

$$f(\mathbf{I}; w) = \mathbf{B}_{s(\mathbf{I};w)} \mathbf{I} + a_{s(\mathbf{I};w)} \tag{10.22}$$

is piecewise linear, where

$$\mathbf{B}_s = \prod_{l=L}^{1} s_l w_l, \tag{10.23}$$

and a_s can be similarly calculated. $s(\mathbf{I}; w)$ partitions the image space of $\mathbf{I}$ into exponentially many pieces [147] according to the value of $s(\mathbf{I}; w)$. The partition is recursive because $s_l(\mathbf{I}; w)$ depends on $s_{l-1}(\mathbf{I}; w)$. The boundaries between the pieces are all linear. On each piece with $s(\mathbf{I}; w) = s$, where s on the right hand side denotes a particular value of $s(\mathbf{I}; w)$, $f(\mathbf{I}; w)$ is a linear function $f(\mathbf{I}; w) = \mathbf{B}_s\mathbf{I} + a_s$. The binary switches in $s(\mathbf{I}; w)$ reconfigure the linear transformation according to (10.23).

$f(\mathbf{I}; w)$ 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 $f(\mathbf{I}; w)$ can be viewed as a multi-dimensional linear spline. The number of linear pieces is exponential in the number of layers [147]. Such a structure can approximate any continuous non-linear function by a large number of linear pieces.

(3) CART [14] and MARS [53]. 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 $s_l(\mathbf{I}; w)$ for $l = 1, ..., L$. Moreover, MARS also makes use of the hinge function $\max(0, r)$.

For Gaussian reference $q(\mathbf{I})$, the energy function is

$$U(\mathbf{I}; w) = -f(\mathbf{I}; w) + \frac{1}{2\sigma^2}||\mathbf{I}||^2. \tag{10.24}$$

We can continue to use Langevin dynamics (11.34) to sample from $p(\mathbf{I}; w)$.

The parameter w can be learned by the stochastic gradient ascent algorithm [214]

$$w^{(t+1)} = w^{(t)} + \gamma_t \left[\frac{1}{n} \sum_{i=1}^{n} \frac{\partial}{\partial w} f(\mathbf{I}_i; w) - \frac{1}{\tilde{n}} \sum_{i=1}^{\tilde{n}} \frac{\partial}{\partial w} f(\tilde{\mathbf{I}}_i; w) \right], \tag{10.25}$$

where again γ_t is the learning rate, and $\{\tilde{\mathbf{I}}_i, i = 1, ..., \tilde{n}\}$ 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 $p(\mathbf{I}; w)$, 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 $\partial f(\mathbf{I}; w)/\partial \mathbf{I}$. In the learning step, we need to compute $\partial f(\mathbf{I}; w)/\partial w$. 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 15×15 filters with sub-sampling size of 3 pixels. The second layer has 64 5×5 filters with sub-sampling size of 1. The third layer has 30 3×3 filters with sub-sampling size of 1. We learn a model (10.17) for each texture category from a single training image. Figure 10.12 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 224×224, 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 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

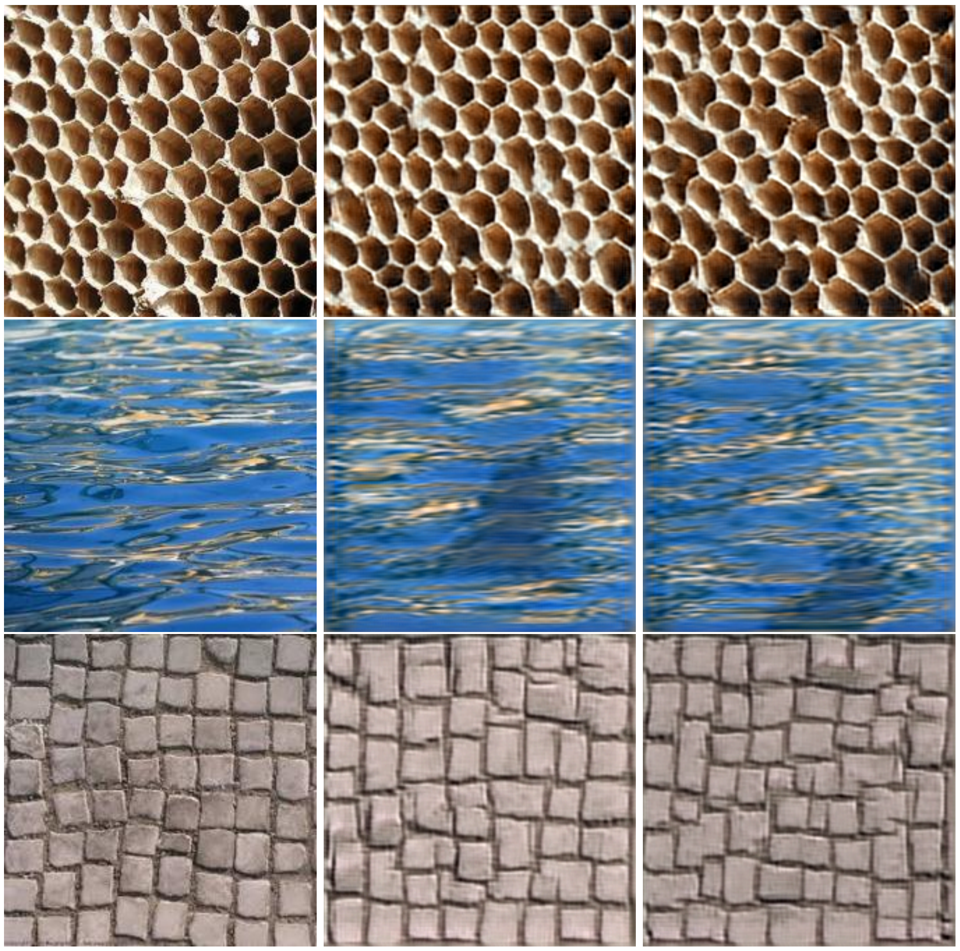


Figure 10.12: Generating texture patterns. For each category, the first image (224 $\times$ 224) is the training image, and the rest are 2 of the images generated by the learning algorithm.

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 [222] 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.

10.1.7 Hopfield auto-encoder

Consider the sparse FRAME model (4.33). 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

$$U(\mathbf{I}) = \frac{1}{2}\|\mathbf{I}\|^2 - \sum_{j=1}^{m} w_j h(\langle \mathbf{I}, B_{k_j,x_j} \rangle). \tag{10.26}$$

This energy function can be multi-modal, and each local minimum $\hat{\mathbf{I}}$ satisfies $U'(\hat{\mathbf{I}}) = 0$. Thus,

$$\hat{\mathbf{I}} = \sum_{j=1}^{m} w_i h'(\langle \hat{\mathbf{I}}, B_{k_j,x_j} \rangle) B_{k_j,x_j}. \tag{10.27}$$

This reveals an auto-encoder [11, 191] hidden in the local modes:

$$\text{Encoding} : c_j = w_j h'(\langle \hat{\mathbf{I}}, B_{k_j,x_j} \rangle), \tag{10.28}$$

$$\text{Decoding} : \hat{\mathbf{I}} = \sum_{i=1}^{n} c_j B_{k_j,x_j}, \tag{10.29}$$

where (10.28) encodes $\hat{\mathbf{I}}$ by (c_j), and (10.29) reconstructs $\hat{\mathbf{I}}$ from (c_j). B_{k_j,x_j} serves as both bottom-up filter in (10.28) and top-down basis function in (10.29). We call this auto-encoder the Hopfield auto-encoder because $\hat{\mathbf{I}}$ is a local minimum of the energy function (10.26). Hopfield [84] 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 (10.17) [208]. The energy function of the model is $\|\mathbf{I}\|^2/2 - f(\mathbf{I}; w)$. The local minima satisfies the Hopfield auto-encoder $\hat{\mathbf{I}} = f'(\hat{\mathbf{I}}; w)$, or more specifically,

$$\text{Encoding} : s = s(\hat{\mathbf{I}}; w), \tag{10.30}$$

$$\text{Decoding} : \hat{\mathbf{I}} = \mathbf{B}_s. \tag{10.31}$$

The encoding process is a bottom-up computation of the indicators at different layers $s_l = s_l(\mathbf{I}; w)$, for $l = 1, ..., L$, where w_l plays the role of filters, see equation (10.21). 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 (10.23). The encoding process detects the patterns corresponding to the filters, and then decoding process reconstructs the image using the detected filters as the basis functions.

The relationship between auto-encoders and energy-based models [112] has been investigated by [190] and [178] for the restricted Boltzmann machine and its extensions [80]. A regularized auto-encoder is a special form of score matching estimator [87]. The Hopfield auto-encoder was first elucidated by [208].

In order to learn the parameters from training images, we may fit the Hopfield auto-encoder using the least squares reconstruction loss. After learning by auto-encoder, we may use MCMC-based learning to further refine the learning results, i.e., learn to synthesize after learning to reconstruct.

10.1.8 Contrastive Divergence and Hopfield Memory

If we want to learn from big data, we may use the contrastive divergence method [78] by starting the Langevin dynamics from the observed images, and run one or a small number of iterations. Then we use the sampled images to approximate the expectation in equation (**??**) for updating the parameters. The contrastive divergence algorithm has become very popular for learning generative models and often leads to reasonable performance, although its theoretical properties are still not well understood. We shall show that the behavior of the contrastive divergence learning of the generative ConvNet is connected to reconstruction.

Suppose we start from an observed image $\mathbf{I}^{\text{obs}}$, and run a small number of iterations of Langevin dynamics (11.34) to get a synthesized image $\mathbf{I}^{\text{syn}}$. If both $\mathbf{I}^{\text{obs}}$ and $\mathbf{I}^{\text{syn}}$ share the same activation pattern $\delta(\mathbf{I}^{\text{obs}}; w) = \delta(\mathbf{I}^{\text{syn}}; w) = \delta$, then $f(\mathbf{I}; w) = a_{w,\delta} + \langle \mathbf{I}, \mathbf{B}_{w,\delta} \rangle$ for both $\mathbf{I}^{\text{obs}}$ and $\mathbf{I}^{\text{syn}}$. Then the contribution of $\mathbf{I}^{\text{obs}}$ to the learning gradient is

$$\frac{\partial}{\partial w} f(\mathbf{I}^{\text{obs}}; w) - \frac{\partial}{\partial w} f(\mathbf{I}^{\text{syn}}; w) = \langle \mathbf{I}^{\text{obs}} - \mathbf{I}^{\text{syn}}, \frac{\partial}{\partial w} \mathbf{B}_{w,\delta} \rangle. \tag{10.32}$$

If $\mathbf{I}^{\rm syn}$ is close to the mean $\mathbf{B}_{w,\delta}$ and if $\mathbf{B}_{w,\delta}$ is a local mode, then the contrastive divergence tends to reconstruct $\mathbf{I}^{\rm obs}$ by the local mode $\mathbf{B}_{w,\delta}$, because the gradient

$$\frac{\partial}{\partial w}\|\mathbf{I}^{\rm obs} - \mathbf{B}_{w,\delta}\|^2/2 = -\langle \mathbf{I}^{\rm obs} - \mathbf{B}_{w,\delta}, \frac{\partial}{\partial w}\mathbf{B}_{w,\delta}\rangle. \tag{10.33}$$

Hence the contrastive divergence learns the Hopfield network which memorizes the observations by the local modes.

We can establish a precise connection for one-step contrastive divergence.

Proposition 8. *Contrastive divergence learns to reconstruct: If the one-step Langevin dynamics does not change the activation pattern, i.e.,* $\delta(\mathbf{I}^{\rm obs}; w) = \delta(\mathbf{I}^{\rm syn}; w) = \delta$, *then the one-step contrastive divergence has an expected gradient that is proportional to the reconstruction gradient:*

$$\mathrm{E}\left[\frac{\partial}{\partial w}f(\mathbf{I}^{\rm obs}; w) - \frac{\partial}{\partial w}f(\mathbf{I}^{\rm syn}; w)\right] \propto \frac{\partial}{\partial w}\|\mathbf{I}^{\rm obs} - \mathbf{B}_{w,\delta}\|^2. \tag{10.34}$$

This is because

$$\mathbf{I}^{\rm syn} = \mathbf{I}^{\rm obs} - \frac{\epsilon^2}{2}\left[\mathbf{I}^{\rm obs} - \mathbf{B}_{w,\delta}\right] + \epsilon Z, \tag{10.35}$$

hence

$$\mathrm{E}_Z\left[\mathbf{I}^{\rm obs} - \mathbf{I}^{\rm syn}\right] \propto \mathbf{I}^{\rm obs} - \mathbf{B}_{w,\delta}, \tag{10.36}$$

and Proposition 8 follows from (10.32) and (10.33).

Proposition 8 is related to score matching estimator of [87], whose connection with the contrastive divergence based on one-step Langevin dynamics was studied by [88]. The relationship between score matching and auto-encoder was discovered by [190] and [178]. Our work can be considered a sharpened specialization of the above mentioned connection and relationship, where the piecewise linear structure of the ConvNet greatly simplifies the matter by getting rid of the complicated second derivative terms, so that the contrastive divergence gradient becomes exactly proportional to the gradient of the reconstruction error, which is not the case in general score matching estimator. Also, our work gives a novel hierarchical realization of the relationship between probability model and auto-encoder, as well as an explicit hierarchical realization of auto-encoder based sampling of [2].

10.1.9 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 [78], 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 [87, 88] and auto-encoder [2, 178, 190]. 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 [182], 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 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.

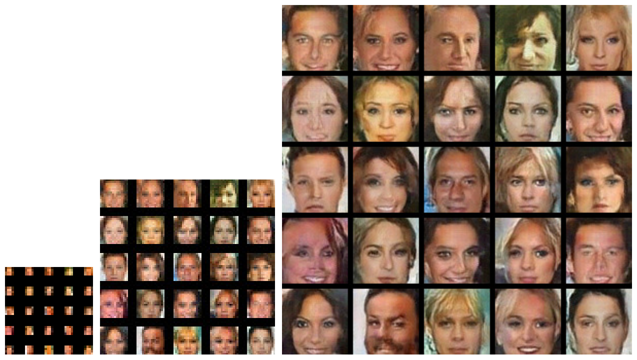


Figure 10.13: Synthesized images at multi-grids [56]. From left to right: 4×4 grid, 16×16 grid and 64×64 grid. Synthesized image at each grid is obtained by 30 step Langevin sampling initialized from the synthesized image at the previous coarser grid, beginning with the 1×1 grid.

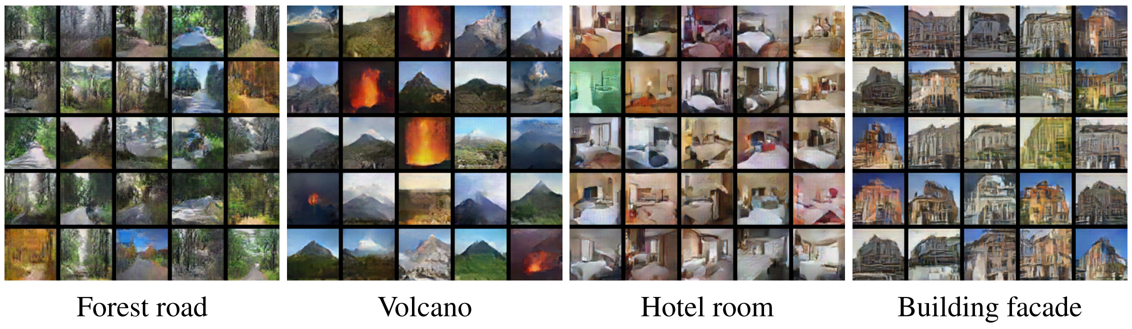


Figure 10.14: Synthesized images from models learned by multi-grid method [56] from 4 categories of MIT places205 datasets.

In [56], we developed a multi-grid sampling and learning method to address the above challenges under the constraint of finite budget MCMC. Specifically, we repeatedly down-scale each training image to get its multi-grid versions. Our method learns a separate descriptive model at each grid. Within each iteration of our learning algorithm, for each observed training image, we generate the corresponding synthesized images at multiple grids. Specifically, we initialize the finite-step MCMC sampling from the minimal 1×1 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 10.13 for an illustration, where we sample images 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 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 1×1 versions of the training images.

In our experiments, the training images are resized to 64×64. Since the models of the three grids act on images of different scales, we design a specific ConvNet structure per grid: grid1 has a 3-layer network with 5×5 stride 2 filters at the first layer and 3×3 stride 1 filters at the next two layers; grid2 has a 4-layer network with 5×5 stride 2 filters at the first layer and 3×3 stride 1 filters at the next three layers; grid3 has a 3-layer network with 5×5 stride 2 filters at the first layer, 3×3 stride 2 filters at the second layer, and 3×3 stride 1 filters at the third layer. Numbers of channels are $96-128-256$ at grid1 and grid3, and $96-128-256-512$ at grid2. A fully-connected layer with 1 channel output is added on top of every grid to get the value of the function $f_\theta(X)$. At each iteration, we run $l = 30$ steps of Langevin dynamics for each grid with step size $s = 0.3$. All networks are trained simultaneously with mini-batches of size 100 and an initial learning rate of 0.3. Learning rate is decayed logarithmically every 10 iterations.

We learn multi-grid models from several datasets including CelebA [120], MIT places205 [219] and CIFAR-10 [108]. In the CelebA dataset, we randomly sample 10,000 images for training. Figure 10.13 shows the synthesized examples. Figure 10.14 shows synthesized images from models learned from 4 categories of MIT places205 dataset by multi-grid method. We learn from each category separately. The number of training images is $15,100$ for each category.

Traditionally, the mixing time of Markov chain is defined via $d(t) = \max_x \|P^{(t)}(x, \cdot) - \pi\|_{\rm TV}$, where $P^{(t)}$ is the t-step transition, π is the stationary distribution, and $\|\cdot\|_{\rm TV}$ is the total variation distance. This is the worst case scenario by choosing the least favorable point mass at x. In our method, however, the initial distribution at each grid can be much more favorable, e.g., it may already agree approximately with π on the marginal distribution of the coarser grid, so that after t steps, the distribution of the sampled image can be close to π, even if this is not the case for the worst case starting point. Such non-persistent finite budget MCMC is computationally more manageable than persistent chains in learning.

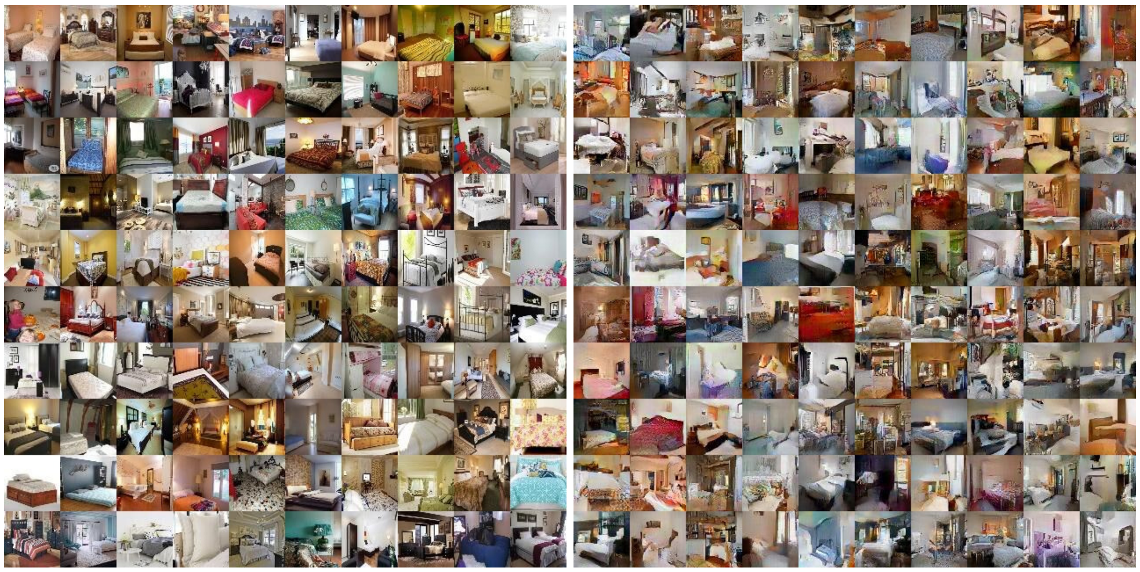


Figure 10.15: Learning the multi-grid models from the LSUN bedroom dataset [56]. Left: random samples of training examples. Right: synthesized examples generated by the learned models.

To train multi-grid models on 10,000 training images for 400 iterations with a singe Titan X GPU, it takes about 7.45 hours. After training, it takes less than 1 second to generate a batch of 100 images. We also train the multi-grid models on LSUN bedroom dataset [215], which consists of roughly 3 million images. Figure 10.15 shows the learning results after 8 epochs.

The learned descriptive 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

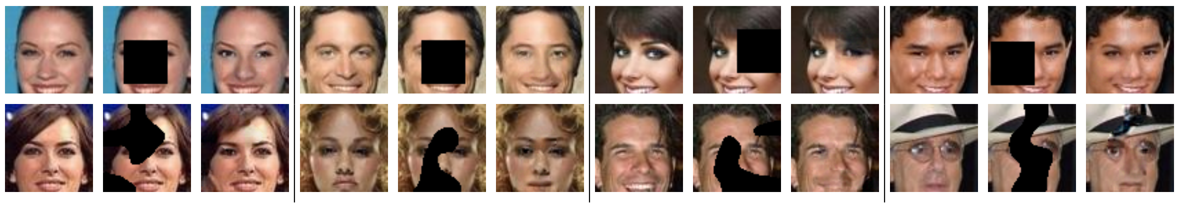


Figure 10.16: Inpainting examples on CelebA dataset [56]. In each block from left to right: the original image; masked input; inpainted image by multi-grid method.

prior distribution for inpainting, as illustrated by Figure 10.16. See [56] for experiment details and numerical evaluations.

10.1.10 Adversarial interpretation

The deep convolutional energy-based model (10.17) can be written as

$$p(\mathbf{I}; w) = \frac{1}{Z(w)} \exp[-U(\mathbf{I}; w)], \tag{10.37}$$

where the energy function $U(\mathbf{I}; w) = -f(\mathbf{I}; w) + \frac{1}{2\sigma^2}||\mathbf{I}||^2$. The update of w is based on $L'(w)$ which can be approximated by

$$\frac{1}{\tilde{n}} \sum_{i=1}^{\tilde{n}} \frac{\partial}{\partial w} U(\tilde{\mathbf{I}}_i; w) - \frac{1}{n} \sum_{i=1}^{n} \frac{\partial}{\partial w} U(\mathbf{I}_i; w), \tag{10.38}$$

where $\{\tilde{\mathbf{I}}_i, i = 1, ..., \tilde{n}\}$ are the synthetic images that are generated by the Langevin dynamics. At the zero temperature limit, the Langevin dynamics becomes gradient descent:

$$\tilde{\mathbf{I}}_{\tau+1} = \tilde{\mathbf{I}}_\tau - \delta \frac{\partial}{\partial \tilde{\mathbf{I}}} U(\tilde{\mathbf{I}}_\tau; w). \tag{10.39}$$

Consider the value function

$$V(\tilde{\mathbf{I}}_i, i = 1, ..., \tilde{n}; w) = \frac{1}{\tilde{n}} \sum_{i=1}^{\tilde{n}} U(\tilde{\mathbf{I}}_i; w) - \frac{1}{n} \sum_{i=1}^{n} U(\mathbf{I}_i; w). \tag{10.40}$$

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 $\{\tilde{\mathbf{I}}_i, i = 1, ..., \tilde{n}\}$ 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 [196] from exponential family models to general energy-based models.

If we recruit a generator model $g(X; \tilde{w})$ as a sampler, then the energy-based model and the generator model play a minimax game with the value function

$$V(\tilde{w}; w) = \mathcal{E}_{\tilde{w}}[U(\tilde{\mathbf{I}}; w)] - \mathcal{E}_{\text{data}}[U(\mathbf{I}; w)], \tag{10.41}$$

where $\mathcal{E}_{\tilde{w}}$ is the expectation with respect to the generator model with parameter $\tilde{w}$, and $\mathcal{E}_{\text{data}}[U(\mathbf{I}; w)] = \frac{1}{n}\sum_{i=1}^{n} U(\mathbf{I}_i; w)$. This is related to [5].

10.1.11 Short-run MCMC

We investigate a learning scheme that is apparently wrong with no hope of learning a valid model. Within each learning iteration, we run a non-convergent, non-mixing and non-persistent short-run MCMC, such as 5 to 100 steps of Langevin dynamics, toward the current EBM. Here, we always initialize the non-persistent short-run MCMC from the same distribution, such as the uniform noise distribution, and we always run the same number of MCMC steps. We then update the model parameters as usual, as if the synthesized examples generated by the non-convergent and non-persistent noise-initialized short-run MCMC are the fair samples generated from the current EBM. We show that, after the convergence of such a learning algorithm, the resulting noise-initialized short-run MCMC can generate realistic images, see Figures 10.17 and 10.18.

The short-run MCMC is not a valid sampler of the EBM because it is short-run. As a result, the learned EBM cannot be a valid model because it is learned based on a wrong sampler. Thus we learn a wrong sampler of a wrong model. However, the short-run MCMC can indeed generate realistic images. What is going on?

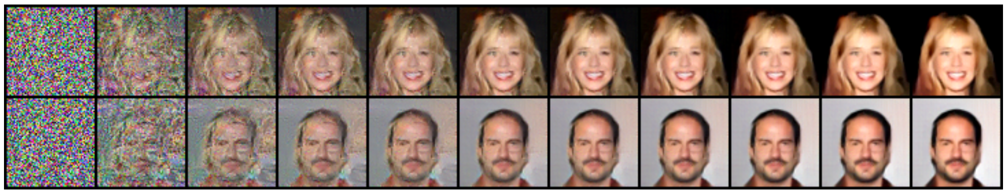


Figure 10.17: **Synthesis by short-run MCMC**: Generating synthesized examples by running 100 steps of Langevin dynamics initialized from uniform noise for CelebA (64×64).

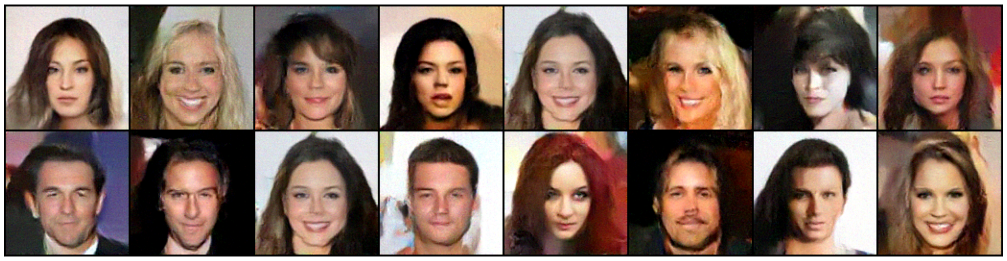


Figure 10.18: **Synthesis by short-run MCMC**: Generating synthesized examples by running 100 steps of Langevin dynamics initialized from uniform noise for CelebA (128×128).

In the following, we will understand the learned short-run MCMC. We provide arguments that it is a valid model for the data in terms of matching the statistical properties of the data distribution. We also show that the learned short-run MCMC can be used as a generative model, such as a generator model [64, 103] or the flow model [8, 36, 37, 65, 104], with the Langevin dynamics serving as a noise-injected residual network, with the initial image serving as the latent variables, and with the initial uniform noise distribution serving as the prior distribution of the latent variables. We show that unlike traditional EBM and MCMC, the learned short-run MCMC is capable of reconstructing the observed images and interpolating different images, just like a generator or a flow model can do. See Figures 10.19 and 10.20. This is very unconventional for EBM or MCMC, and this is due to the fact that the MCMC is non-convergent, non-mixing and non-persistent. In fact, our argument applies to the situation where the short-MCMC does not need to have the EBM as the stationary distribution.

This perspective constitutes a conceptual shift, where we shift attention from learning EBM with unrealistic convergent MCMC to the non-convergent short-run MCMC. This is a break away from the long tradition of both EBM and MCMC. We provide theoretical and empirical evidence that the learned short-run MCMC is a valid generator or flow model. This conceptual shift frees us from the convergence issue of MCMC, and makes the short-run MCMC a reliable and efficient technology.

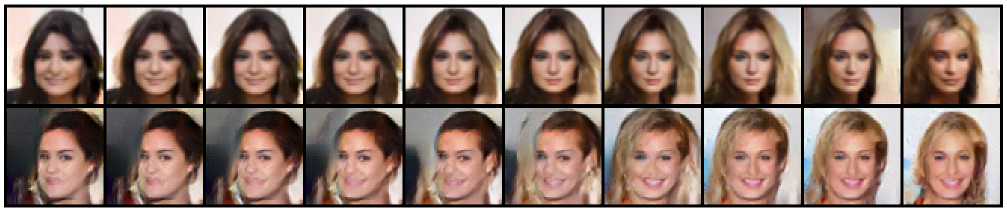


Figure 10.19: **Interpolation by short-run MCMC resembling a generator or flow model**: The transition depicts the sequence $M_\theta(z_\rho)$ with interpolated noise $z_\rho = \rho z_1 + \sqrt{1-\rho^2} z_2$ where $\rho \in [0,1]$ on CelebA (64×64). Left: $M_\theta(z_1)$. Right: $M_\theta(z_2)$.

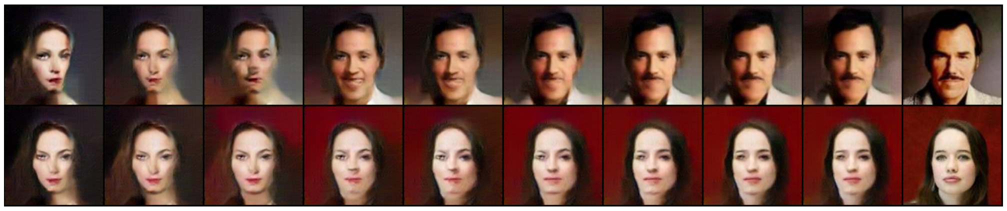


Figure 10.20: **Reconstruction by short-run MCMC resembling a generator or flow model**: The transition depicts $M_\theta(z_t)$ over time t from random initialization $t = 0$ to reconstruction $t = 200$ on CelebA (64×64). Left: Random initialization. Right: Observed examples.

More generally, we shift the focus from energy-based model to energy-based dynamics. This appears to be consistent with the common practice of computational neuroscience [110], where researchers often directly start from the dynamics, such as attractor dynamics [4, 84, 149] whose express goal is to be trapped in a local mode. It is our hope that our work may help to understand the learning of such dynamics. We leave it to future work.

For short-run MCMC, contrastive divergence (CD) [78] is the most prominent framework for theoretical underpinning. The difference between CD and our study is that in our study, the short-run MCMC is initialized from noise, while CD initializes from observed images. CD has been generalized to persistent CD [183]. Compared to persistent MCMC, the non-persistent MCMC in our method is much more efficient and convenient. [141] performs a thorough investigation of various persistent and non-persistent, as well as convergent and non-convergent learning schemes. In particular, the emphasis is on learning proper energy function with persistent and convergent Markov chains. In all of the CD-based frameworks, the goal is to learn the EBM, whereas in our framework, we discard the learned EBM, and only keep the learned short-run MCMC.

Let x be the signal, such as an image. The energy-based model (EBM) is a Gibbs distribution

$$p_\theta(x) = \frac{1}{Z(\theta)} \exp(f_\theta(x)), \tag{10.42}$$

where we assume x is within a bounded range. $f_\theta(x)$ is the negative energy and is parametrized by a bottom-up convolutional neural network (ConvNet) with weights θ. $Z(\theta) = \int \exp(f_\theta(x))dx$ is the normalizing constant.

Suppose we observe training examples $x_i, i = 1, ..., n \sim P_{\text{data}}$, where P_{data} is the data distribution. For large n, the sample average over $\{x_i\}$ approximates the expectation with respect with P_{data}. For notational convenience, we treat the sample average and the expectation as the same.

The log-likelihood is

$$L(\theta) = \frac{1}{n}\sum_{i=1}^{n} \log p_\theta(x_i) \doteq \mathcal{E}_{P_{\text{data}}}[\log p_\theta(x)]. \tag{10.43}$$

The derivative of the log-likelihood is

$$L'(\theta) = \mathcal{E}_{P_{\text{data}}}\left[\dot{\text{f}}_\theta(x)\right] - \mathcal{E}_{p_\theta}\left[\dot{\text{f}}_\theta(x)\right] \doteq \frac{1}{n}\sum_{i=1}^{n}\dot{\text{f}}_\theta(x_i) - \frac{1}{n}\sum_{i=1}^{n}\dot{\text{f}}_\theta(x_i^-), \tag{10.44}$$

where $x_i^- \sim p_\theta(x)$ for $i = 1, ..., n$ 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 θ_t be the current model parameters. We generate $x_i^- \sim p_{\theta_t}(x)$ for $i = 1, ..., n$. Then we update $\theta_{t+1} = \theta_t + \eta_t L'(\theta_t)$, where η_t is the learning rate.

Generating synthesized examples $x_i^- \sim p_\theta(x)$ requires MCMC, such as Langevin dynamics (or Hamiltonian Monte Carlo) [138], which iterates

$$x_{\tau+\Delta\tau} = x_\tau + \frac{\Delta\tau}{2} f'_\theta(x_\tau) + \sqrt{\Delta\tau}U_\tau, \tag{10.45}$$

where τ indexes the time, $\Delta\tau$ is the discretization of time, and $U_\tau \sim \text{N}(0, I)$ is the Gaussian noise term. $f'_\theta(x) = \partial f_\theta(x)/\partial x$ can be obtained by back-propagation. If p_θ is of low entropy or low temperature, the gradient term dominates the diffusion noise term, and the Langevin dynamics behaves like gradient descent.

If $f_\theta(x)$ is multi-modal, then different chains tend to get trapped in different local modes, and they do not mix. We propose to give up the sampling of p_θ. Instead, we run a fixed number, e.g., K, steps of MCMC, toward p_θ, starting from a fixed initial distribution, p_0, such as the uniform noise distribution. Let M_θ be the K-step MCMC transition kernel. Define

$$q_\theta(x) = (M_\theta p_0)(z) = \int p_0(z) M_\theta(x|z)dz, \tag{10.46}$$

which is the marginal distribution of the sample x after running K-step MCMC from p_0.

In this paper, instead of learning p_θ, we treat q_θ to be the target of learning. After learning, we keep q_θ, but we discard p_θ. That is, the sole purpose of p_θ is to guide a K-step MCMC from p_0.

The learning algorithm is as follows. Initialize θ_0. At learning iteration t, let θ_t be the model parameters. We generate $x_i^- \sim q_{\theta_t}(x)$ for $i = 1, ..., m$. Then we update $\theta_{t+1} = \theta_t + \eta_t \Delta(\theta_t)$, where

$$\Delta(\theta) = \mathcal{E}_{P_{\text{data}}}\left[\dot{\text{f}}_\theta(x)\right] - \mathcal{E}_{q_\theta}\left[\dot{\text{f}}_\theta(x)\right] \approx \sum_{i=1}^{m}\dot{\text{f}}_\theta(x_i) - \sum_{i=1}^{m}\dot{\text{f}}_\theta(x_i^-). \tag{10.47}$$

We assume that the algorithm converges so that $\Delta(\theta_t) \to 0$. At convergence, the resulting θ solves the estimating equation $\Delta(\theta) = 0$.

To further improve training, we smooth P_{data} by convolution with a Gaussian white noise distribution, i.e., injecting additive noises $\epsilon_i \sim \text{N}(0, \sigma^2 I)$ to observed examples $x_i \leftarrow x_i + \epsilon_i$ [157, 175]. This makes it easy for $\Delta(\theta_t)$ to converge to 0, especially if the number of MCMC steps, K, is small, so that the estimating equation $\Delta(\theta) = 0$ may not have solution without smoothing P_{data}.

We may consider $q_\theta(x)$ to be a generative model,

$$z \sim p_0(z);\ x = M_\theta(z, u), \tag{10.48}$$

where u denotes all the randomness in the short-run MCMC. For the K-step Langevin dynamics, M_θ can be considered a K-layer noise-injected residual network. z can be considered latent variables, and p_0 the prior distribution of z. Due to the non-convergence and non-mixing, x can be highly dependent on z, and z can be inferred from x. This is different from the convergent MCMC, where x is independent of z. When the learning algorithm converges, the learned EBM tends to have low entropy and the Langevin dynamics behaves like gradient descent, where the noise terms are disabled, i.e., $u = 0$. In that case, we simply write $x = M_\theta(z)$.

We can perform interpolation as follows. Generate z_1 and z_2 from $p_0(z)$. Let $z_\rho = \rho z_1 + \sqrt{1-\rho^2} z_2$. This interpolation keeps the marginal variance of z_ρ fixed. Let $x_\rho = M_\theta(z_\rho)$. Then x_ρ is the interpolation of $x_1 = M_\theta(z_1)$ and $x_2 = M_\theta(z_2)$. Figure 10.19 displays x_ρ for a sequence of $\rho \in [0, 1]$.

For an observed image x, we can reconstruct x by running gradient descent on the least squares loss function $L(z) = \|x - M_\theta(z)\|^2$, initializing from $z_0 \sim p_0(z)$, and iterates $z_{t+1} = z_t - \eta_t L'(z_t)$. Figure 10.20 displays the sequence of $x_t = M_\theta(z_t)$.

In general, $z \sim p_0(z); x = M_\theta(z, u)$ defines an energy-based dynamics. K does not need to be fixed. It can be a stopping time that depends on the past history of the dynamics. The dynamics can be made deterministic by setting $u = 0$. This includes the attractor dynamics popular in computational neuroscience [4, 84, 149].

An early version of EBM is the FRAME (Filters, Random field, And Maximum Entropy) model [203, 221, 225], which is an exponential family model, where the features are the responses from a bank of filters. The deep FRAME model [122] replaces the linear filters by the pre-trained ConvNet filters. This amounts to only learning the top layer weight parameters of the ConvNet. Specifically, $f_\theta(x) = \langle \theta, h(x) \rangle$, where $h(x)$ are the top-layer filter responses of a pre-trained ConvNet, and θ consists of the top-layer weight parameters. For such an $f_\theta(x)$, $\dot{f}_\theta(x) = h(x)$. Then, the maximum likelihood estimator of p_θ is actually a moment matching estimator, i.e., $\mathcal{E}_{p_{\hat{\theta}_{\text{MLE}}}}[h(x)] = \mathcal{E}_{P_{\text{data}}}[h(x)]$. If we use the short-run MCMC learning algorithm, it will converge (assume convergence is attainable) to a moment matching estimator, i.e., $\mathcal{E}_{q_{\hat{\theta}_{\text{MME}}}}[h(x)] = \mathcal{E}_{P_{\text{data}}}[h(x)]$. Thus, the learned model $q_{\hat{\theta}_{\text{MME}}}(x)$ is a valid estimator in that it matches to the data distribution in terms of sufficient statistics defined by the EBM.

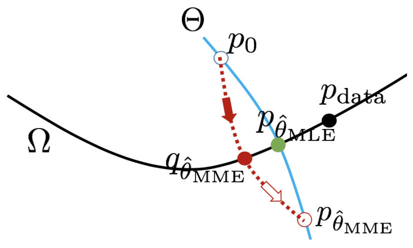

Θ
p_0
p_{data}
$p_{\hat{\theta}_{\text{MLE}}}$
Ω
$q_{\hat{\theta}_{\text{MME}}}$
$p_{\hat{\theta}_{\text{MME}}}$

Figure 10.21: The blue curve illustrates the model distributions corresponding to different values of parameter θ. The black curve illustrates all the distributions that match P_{data} (black dot) in terms of $\mathcal{E}[h(x)]$. The MLE $p_{\hat{\theta}_{\text{MLE}}}$ (green dot) is the intersection between Θ (blue curve) and Ω (black curve). The MCMC (red dotted line) starts from p_0 (hollow blue dot) and runs toward $p_{\hat{\theta}_{\text{MME}}}$ (hollow red dot), but the MCMC stops after K-step, reaching $q_{\hat{\theta}_{\text{MME}}}$ (red dot), which is the learned short-run MCMC.

Consider two families of distributions: $\Omega = \{p : \mathcal{E}_p[h(x)] = \mathcal{E}_{P_{\text{data}}}[h(x)]\}$, and $\Theta = \{p_\theta(x) = \exp(\langle \theta, h(x) \rangle)/Z(\theta), \forall \theta\}$. They are illustrated by two curves in Figure 10.21. Ω contains all the distributions that match the data distribution in terms of $\mathcal{E}[h(x)]$. Both $p_{\hat{\theta}_{\rm MLE}}$ and $q_{\hat{\theta}_{\rm MME}}$ belong to Ω, and of course $P_{\rm data}$ also belongs to Ω. Θ contains all the EBMs with different values of the parameter θ. The uniform distribution p_0 corresponds to $\theta = 0$, thus p_0 belongs to Θ.

The EBM under $\hat{\theta}_{\rm MME}$, i.e., $p_{\hat{\theta}_{\rm MME}}$ does not belong to Ω, and it may be quite far from $p_{\hat{\theta}_{\rm MLE}}$. In general, $\mathcal{E}_{p_{\hat{\theta}_{\rm MME}}}[h(x)] \neq \mathcal{E}_{P_{\rm data}}[h(x)]$, that is, the corresponding EBM does not match the data distribution as far as $h(x)$ is concerned. It can be much further from the uniform p_0 than $p_{\hat{\theta}_{\rm MLE}}$ is from p_0, and thus $p_{\hat{\theta}_{\rm MME}}$ may have a much lower entropy than $p_{\hat{\theta}_{\rm MLE}}$.

Figure 10.21 illustrates the above idea. The red dotted line illustrates MCMC. Starting from p_0, K-step MCMC leads to $q_{\hat{\theta}_{\rm MME}}(x)$. If we continue to run MCMC for infinite steps, we will get to $p_{\hat{\theta}_{\rm MME}}$. Thus the role of $p_{\hat{\theta}_{\rm MME}}$ is to serve as an unreachable target to guide the K-step MCMC which stops at the mid-way $q_{\hat{\theta}_{\rm MME}}$. One can say that the short-run MCMC is a wrong sampler of a wrong model, but it itself is a valid model because it belongs to Ω.

The MLE $p_{\hat{\theta}_{\rm MLE}}$ is the projection of $P_{\rm data}$ onto Θ. Thus it belongs to Θ. It also belongs to Ω as can be seen from the maximum likelihood estimating equation. Thus it is the intersection of Ω and Θ. Among all the distributions in Ω, $p_{\hat{\theta}_{\rm MLE}}$ is the closest to p_0. Thus it has the maximum entropy among all the distributions in Ω.

The above duality between maximum likelihood and maximum entropy follows from the following fact. Let $\hat{p} \in \Theta \cap \Omega$ be the intersection between Θ and Ω. Ω and Θ are orthogonal in terms of the Kullback-Leibler divergence. For any $p_\theta \in \Theta$ and for any $p \in \Omega$, we have the Pythagorean property [**?**]: ${\rm KL}(p|p_\theta) = {\rm KL}(p|\hat{p}) + {\rm KL}(\hat{p}|p_\theta)$. Thus (1) ${\rm KL}(P_{\rm data}|p_\theta) \geq {\rm KL}(P_{\rm data}|\hat{p})$, i.e., $\hat{p}$ is MLE within Θ. (2) ${\rm KL}(p|p_0) \geq {\rm KL}(\hat{p}|p_0)$, i.e., $\hat{p}$ has maximum entropy within Ω.

We can understand the learned $q_{\hat{\theta}_{\rm MME}}$ from two Pythagorean results.

(1) Pythagorean for the right triangle formed by q_0, $q_{\hat{\theta}_{\rm MME}}$, and $p_{\hat{\theta}_{\rm MLE}}$,

$$
{\rm KL}(q_{\hat{\theta}_{\rm MME}}|p_{\hat{\theta}_{\rm MLE}}) = {\rm KL}(q_{\hat{\theta}_{\rm MME}}|p_0) - {\rm KL}(p_{\hat{\theta}_{\rm MLE}}|p_0) = H(p_{\hat{\theta}_{\rm MLE}}) - H(q_{\hat{\theta}_{\rm MME}}), \tag{10.49}
$$

where $H(p) = -\mathcal{E}_p[\log p(x)]$ is the entropy of p. Thus we want the entropy of $q_{\hat{\theta}_{\rm MME}}$ to be high in order for it to be a good approximation to $p_{\hat{\theta}_{\rm MLE}}$. Thus for small K, it is important to let p_0 be the uniform distribution, which has the maximum entropy.

(2) Pythagorean for the right triangle formed by $p_{\hat{\theta}_{\rm MME}}$, $q_{\hat{\theta}_{\rm MME}}$, and $p_{\hat{\theta}_{\rm MLE}}$,

$$
{\rm KL}(q_{\hat{\theta}_{\rm MME}}|p_{\hat{\theta}_{\rm MME}}) = {\rm KL}(q_{\hat{\theta}_{\rm MME}}|p_{\hat{\theta}_{\rm MLE}}) + {\rm KL}(p_{\hat{\theta}_{\rm MLE}}|p_{\hat{\theta}_{\rm MME}}). \tag{10.50}
$$

For fixed θ, as K increases, ${\rm KL}(q_\theta|p_\theta)$ decreases monotonically [26]. The smaller ${\rm KL}(q_{\hat{\theta}_{\rm MME}}|p_{\hat{\theta}_{\rm MME}})$ is, the smaller ${\rm KL}(q_{\hat{\theta}_{\rm MME}}|p_{\hat{\theta}_{\rm MLE}})$ and ${\rm KL}(p_{\hat{\theta}_{\rm MLE}}|p_{\hat{\theta}_{\rm MME}})$ are. Thus, it is desirable to use large K as long as we can afford the computational cost, to make both $q_{\hat{\theta}_{\rm MME}}$ and $p_{\hat{\theta}_{\rm MME}}$ close to $p_{\hat{\theta}_{\rm MLE}}$.

10.2 Generator Network

This section studies the fundamental problem of learning and inference in the generator network [64]. 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.

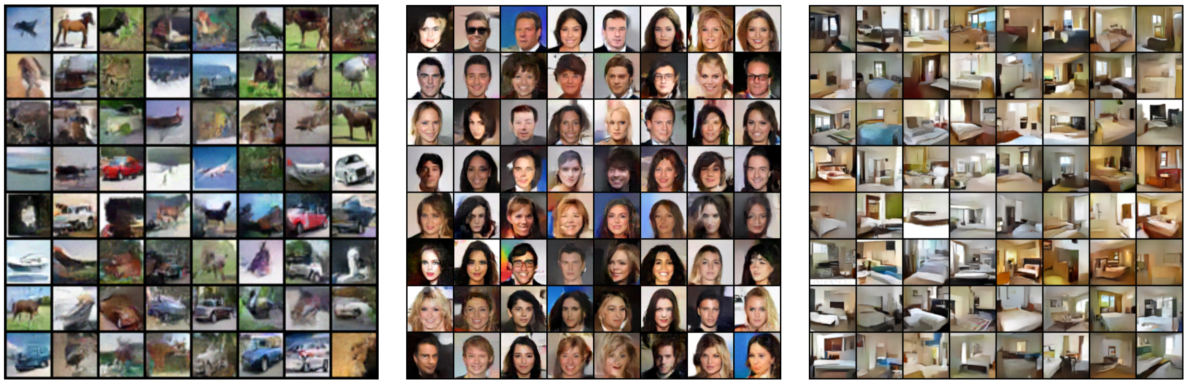


Figure 10.22: Generated samples for $K = 100$ MCMC steps. From left to right: (1) CIFAR-10 (32×32), (2) CelebA (64×64), (3) LSUN Bedroom (64×64).

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 overhead in computing. The inferential back-propagation seeks to perform accurate explaining-away inference of the latent factors.

10.2.1 Factor analysis

Let Y be a D-dimensional observed data vector, such as an image. Let X be the d-dimensional vector of continuous latent factors, $X = (x_k, k = 1, ..., d)$. The traditional factor analysis model is $Y = WX + \epsilon$, where W is $D \times d$ matrix, and ϵ is a D-dimensional error vector or the observational noise. We assume that $X \sim \mathrm{N}(0, I_d)$, where I_d stands for the d-dimensional identity matrix. We also assume that $\epsilon \sim \mathrm{N}(0, \sigma^2 I_D)$, i.e., the observational errors are Gaussian white noises. There are three perspectives to view W.

(1) *Basis vectors*. Write $W = (W_1, ..., W_d)$, where each W_k is a D-dimensional column vector. Then $Y = \sum_{k=1}^{d} x_k W_k + \epsilon$, i.e., W_k are the basis vectors and z_k are the coefficients.

(2) *Loading matrix*. Write $W = (w_1, ..., w_D)^\top$, where $w_j^\top$ is the j-th row of W. Then $y_j = \langle w_j, X\rangle + \epsilon_j$, where y_j and ϵ_j are the j-th components of Y and ϵ respectively. Each y_j is a loading of the d factors where w_j is a vector of loading weights, indicating which factors are important for determining y_j. W is called the loading matrix.

(3) *Matrix factorization*. Suppose we observe $\mathbf{Y} = (Y_1, ..., Y_n)$, whose factors are $\mathbf{X} = (X_1, ..., X_n)$, then $\mathbf{Y} \approx W\mathbf{X}$.

The factor analysis model can be learned by the Rubin-Thayer EM algorithm, which involves alternating regressions of X on Y in the E-step and of Y on X in the M-step, with both steps powered by the sweep operator [119, 161].

The factor analysis model is the prototype of many subsequent models that generalize the prior model of X.

(1) *Independent component analysis* [89], $d = D$, $\epsilon = 0$, and x_k are assumed to follow independent

heavy tailed distributions.

(2) *Sparse coding* [144], $d > D$, and X is assumed to be a redundant but sparse vector, i.e., only a small number of x_k are non-zero or significantly different from zero.

(3) *Non-negative matrix factorization* [114], it is assumed that $x_k \geq 0$.

10.2.2 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 [89], sparse coding [144], non-negative matrix factorization [114], matrix factorization and completion for recommender systems [106], 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) [40, 109, 111]. It has been shown that the generator network is capable of generating realistic images [34, 150].

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 X, we can also generalize the mapping from X to Y. Here, we consider the generator network model [64] that retains the assumptions that $d < D$, $X \sim \mathrm{N}(0, I_d)$, and $\epsilon \sim \mathrm{N}(0, \sigma^2 I_D)$ as in traditional factor analysis, but generalizes the linear mapping WX to a non-linear mapping $f(X; W)$, where f is a ConvNet, and W collects all the connection weights and bias terms of the ConvNet. Then the model becomes

$$\begin{aligned} &Y = f(Z; W) + \epsilon, \\ &Z \sim \mathrm{N}(0, I_d),\ \epsilon \sim \mathrm{N}(0, \sigma^2 I_D),\ d < D. \end{aligned} \tag{10.51}$$

The reconstruction error is $||Y - f(X; W)||^2$. We may assume more sophisticated models for ϵ, such as colored noise or non-Gaussian texture. If Y is binary, we can emit Y by a probability map $P = 1/[1 + \exp(-f(X; W))]$, where the sigmoid transformation and Bernoulli sampling are carried out pixel-wise. If Y is multi-level, we may assume multinomial logistic emission model or some ordinal emission model.

Although $f(X; W)$ can be any non-linear mapping, the ConvNet parameterization of $f(Z; W)$ makes it particularly close to the original factor analysis. Specifically, we can write the top-down ConvNet as follows:

$$X^{(l-1)} = f_l(W_l X^{(l)} + b_l), \tag{10.52}$$

where f_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 $W = (W_l, b_l, l = 1, ..., L)$. $X^{(0)} = f(X; W)$, and $X^{(L)} = X$. The top-down ConvNet (10.52) can be considered a recursion of the 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.

10.2.3 Learning by alternating back-propagation

The factor analysis model can be learned by the Rubin-Thayer EM algorithm [32, 161], 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 [138] 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 [40] 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 [98, 106], and alternating gradient descent algorithm for sparse coding [144]. 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.

The generator net [64] has its root in factor analysis in statistics and in latent variable model or directed graphical model in machine learning. The generator net seeks to explain the signal Y of dimension D by a vector of latent factors X of dimension d, and usually $d \ll D$. If we observe a training set of data vectors $\{Y_i, i = 1, ..., n\}$, then each Y_i has a corresponding X_i, but all the Y_i share the same ConvNet W. Intuitively, we should infer $\{X_i\}$ and learn W to minimize the reconstruction error $\sum_{i=1}^{n} ||Y_i - f(X_i; W)||^2$ plus a regularization term that corresponds to the prior on X.

More formally, the model can be written as

$$\begin{aligned} &X \sim \mathrm{N}(0, I_d), \\ &Y = g(X; W) + \epsilon,\ \epsilon \sim \mathrm{N}(0, \sigma^2 I_D). \end{aligned} \tag{10.53}$$

$g(X; W)$ (g stands for generator) is a top-down ConvNet defined by the parameters W. The ConvNet g maps the latent factors X to the signal Y. See the diagram in (**??**).

Figure 10.23 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 X using inferential back-propagation, and then reconstruct the testing image by $f(X; W)$ using the inferred X and the learned W.

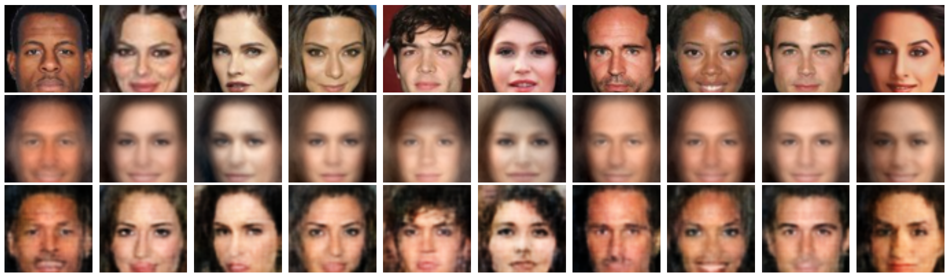


Figure 10.23: Comparison between generator as non-linear and PCA as linear factor analysis. *Row 1:* original testing images. *Row 2:* reconstructions by PCA eigenvectors learned from training images. *Row 3:* reconstructions by the generator learned from training images.

Model (10.53) is a directed graphical model, where Y can be readily generated by first sampling X from its known prior distribution $\mathrm{N}(0, I_d)$ and then transforming X to Y via g. The joint density of model (10.53) is $P(X, Y; W) = P(X)P(Y|X; W)$, and

$$\log P(X, Y; W) = -\frac{1}{2\sigma^2}\|Y - g(X; W)\|^2 - \frac{1}{2}\|X\|^2 + \text{constant},$$

where the constant term is independent of X, Y and W.

The marginal density is obtained by integrating out the latent factors X, i.e., $P(Y; W) = \int P(X, Y; W)dX$. The inference of X given Y is based on the posterior density $P(X|Y; W) = P(X, Y; W)/P(Y; W) \propto P(X, Y; W)$ as a function of X.

For the training data $\{Y_i, i = 1, ..., n\}$, the generator net can be trained by maximizing the log-likelihood

$$L(W) = \frac{1}{n}\sum_{i=1}^{n} \log P(Y_i; W). \tag{10.54}$$

For large sample, W is obtained by minimizing the Kullback-Leibler divergence $\mathrm{KL}(P_{\text{data}}|P)$ from the data distribution P_{data} to the model distribution P.

The gradient of $L(W)$ is obtained according to the following identity

$$\begin{aligned}
\frac{\partial}{\partial W}\log P(Y; W) &= \frac{1}{P(Y; W)}\frac{\partial}{\partial W}\int P(Y, X; W)dX \\
&= \frac{1}{P(Y; W)}\int \left[\frac{\partial}{\partial W}\log P(Y, X; W)\right] P(Y, X; W)dX \\
&= \int \left[\frac{\partial}{\partial W}\log P(Y, X; W)\right] \frac{P(Y, X; W)}{P(Y; W)}dX \\
&= \mathcal{E}_{P(X|Y;W)}\left[\frac{\partial}{\partial W}\log P(X, Y; W)\right]. \qquad (10.55)
\end{aligned}$$

The above identity underlies the EM algorithm, where $\mathcal{E}_{P(X|Y;W)}$ is the expectation with respect to the posterior distribution of the latent factors $P(X|Y; W)$, and is computed in the E-step. The usefulness of identity (10.55) lies in the fact that the derivative of the complete-data log-likelihood $\log P(X, Y; W)$ on the right hand side can be obtained in closed form. In the EM algorithm, the M-step maximizes the expectation of $\log P(X, Y; W)$ with respect to the current posterior distribution of the latent factors.

In general, the expectation in (10.55) is analytically intractable, and has to be approximated by MCMC that samples from the posterior $P_G(X|Y;W)$, such as the Langevin inference dynamics, which iterates

$$X_{\tau+1} = X_\tau + \frac{\delta^2}{2}\frac{\partial}{\partial X}\log P(X_\tau, Y; W) + \delta U_\tau, \tag{10.56}$$

where τ indexes the time step, δ is the step size, and for notational simplicity, we continue to use U_τ to denote the noise term, but here $U_\tau \sim \mathrm{N}(0, I_d)$.

We take the derivative of $\log P(X,Y;W)$ in (10.56) because this derivative is the same as the derivative of the log-posterior $\log P_G(X|Y;W)$, since $P(X|Y;W)$ is proportional to $P(X,Y;W)$ as a function of X.

To explain Langevin dynamics, its continuous time version for sampling $\pi(x) \propto \exp[-\mathcal{E}(x)]$ is $x_{t+\Delta t} = x_t - \Delta t\mathcal{E}'(x_t)/2 + \sqrt{\Delta t}U_t$. The dynamics has π as its stationary distribution, because it can be shown that for any well-behaved testing function h, if $x_t \sim \pi$, then $\mathcal{E}[h(x_{t+\Delta t})] - \mathcal{E}[h(x_t)] \to 0$, as $\Delta t \to 0$, so that $x_{t+\Delta t} \sim \pi$. Alternatively, given $x_t = x$, suppose $x_{t+\Delta t} \sim K(x,y)$, then $[\pi(y)K(y,x)]/[\pi(x)K(x,y)] \to 1$ as $\Delta t \to 0$.

The Langevin inference solves a ℓ_2 penalized non-linear least squares problem so that X_i can reconstruct Y_i given the current W. The Langevin inference process performs explaining-away reasoning, where the latent factors in X compete with each other to explain the current residual $Y - g(X;W)$.

The stochastic gradient algorithm of [214] can be used for learning, where in each iteration, for each X_i, only a single copy of X_i is sampled from $p(X_i|Y_i, W)$ by running a finite number of steps of Langevin dynamics starting from the current value of X_i, i.e., the warm start. With X_i sampled from $P(X_i \mid Y_i, W)$ for each observation Y_i by the Langevin inference process, the Monte Carlo approximation to $L'(W)$ is

$$\begin{aligned} L'(W) &\approx \frac{1}{n}\sum_{i=1}^{n}\frac{\partial}{\partial W}\log P(X_i, Y_i; W) \\ &= \frac{1}{n}\sum_{i=1}^{n}\frac{1}{\sigma^2}\left(Y_i - g(X_i;W)\right)\frac{\partial}{\partial W}g(X_i;W). \end{aligned} \tag{10.57}$$

The updating of W solves a non-linear regression problem, so that the learned W enables better reconstruction of Y_i by the inferred X_i. Given the inferred X_i, the learning of W is a supervised learning problem [40].

Algorithm 7 [72] describes the training algorithm that iterates the following two steps: (1) Inference step: update X_i by running Y steps of gradient descent. (2) Learning step: update W by one step of gradient descent. Step G1 needs to compute $\frac{\partial}{\partial X}g(X;W)$. Step G2 needs to compute $\frac{\partial}{\partial W}g(X;W)$. 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 [214]. If the Gaussian noise U_τ in the Langevin dynamics (10.56) is removed, then the above algorithm becomes the alternating gradient descent algorithm. It is possible to update both W and $\{X_i\}$ simultaneously by joint gradient descent.

Both the inferential back-propagation and the learning back-propagation are guided by the residual $Y_i - f(X_i;W)$. The inferential back-propagation is based on $\partial f(X;W)/\partial X$, whereas the learning back-propagation is based on $\partial f(X;W)/\partial W$. 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 $p(X_i|Y_i, W)$ because W keeps changing. The updating of both X_i and W collaborate to reduce the reconstruction error $||Y_i - f(X_i;W)||^2$. The parameter σ^2 plays the role of annealing or tempering in Langevin sampling. If σ^2 is very large, then the posterior is close to the prior $\mathrm{N}(0, I_d)$. If σ^2 is very small, then the posterior may

```
Input:
    (1) training examples $\{Y_i, i = 1, ..., n\}$
    (2) number of Langevin steps $l$
    (3) number of learning iterations $T$
Output:
    (1) estimated parameters $W$
    (2) inferred latent factors $\{X_i, i = 1, ..., n\}$

1: Let $t \leftarrow 0$, initialize $W$.
2: Initialize $X_i$, $i = 1, ..., n$.
3: repeat
4:     Step G1 Langevin inference: For each $i$, run $l$ steps of Langevin dynamics to update $X_i$, i.e.,
       starting from the current $X_i$, each step follows equation (10.56).
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 (10.57).
6:     Let $t \leftarrow t + 1$
7: until $t = T$
```

Algorithm 7: Algorithm G

be multi-modal, but the evolving energy landscape of $p(X_i|Y_i, W)$ may help alleviate the trapping of the local modes. In practice, we tune the value of σ^2 instead of estimating it. The Langevin dynamics can be extended to Hamiltonian Monte Carlo [138] or more sophisticated versions [63].

Figure 10.24 illustrates the results of modeling textures where we learn a separate model from each texture image. The factors X at the top layer form a $\sqrt{d} \times \sqrt{d}$ image, with each pixel following $\mathrm{N}(0, 1)$ independently. The $\sqrt{d} \times \sqrt{d}$ image X is then transformed to Y by the top-down ConvNet. In order to obtain the synthesized image, we randomly sample a 7×7 image Z from $\mathrm{N}(0, I)$, and then expand the learned network W to generate the 448×448 synthesized image $f(X; W)$.

10.2.4 EM, density mapping, and density shifting

Suppose the training data $\{Y_i, i = 1, ..., n\}$ come from a data distribution $P_{\text{data}}(Y)$. To understand how the alternating back-propagation algorithm or its EM idealization maps the prior distribution of the latent factors $p(X)$ to the data distribution $P_{\text{data}}(Y)$ by the learned $g(X; W)$, we define

$$\begin{aligned} P_{\text{data}}(X, Y; W) &= P_{\text{data}}(Y) p(X|Y, W) \\ &= P_{\text{data}}(X; W) P_{\text{data}}(Y|X, W), \end{aligned} \tag{10.58}$$

where $P_{\text{data}}(X; W) = \int p(X|Y, W) P_{\text{data}}(Y) dY$ is obtained by averaging the posteriors $p(X|Y; W)$ over the observed data $Y \sim P_{\text{data}}$. That is, $P_{\text{data}}(X; W)$ can be considered the data prior. The data prior $P_{\text{data}}(X; W)$ is close to the true prior $p(X)$ in the sense that

$$\begin{aligned} \mathrm{KL}(P_{\text{data}}(X; W)|p(X)) &\leq \mathrm{KL}(P_{\text{data}}(Y)|p(Y; W)) \qquad (10.59) \\ &= \mathrm{KL}(P_{\text{data}}(X, Y; W)|p(X, Y; W)). \end{aligned}$$

The difference between the two sides of (10.59) is $\mathrm{KL}(P_{\text{data}}(Y|X, W)|p(Y|X, W))$. The right hand side of (10.59) is minimized at the maximum likelihood estimate $\hat{W}$, hence the data prior $P_{\text{data}}(X; \hat{W})$ at $\hat{W}$

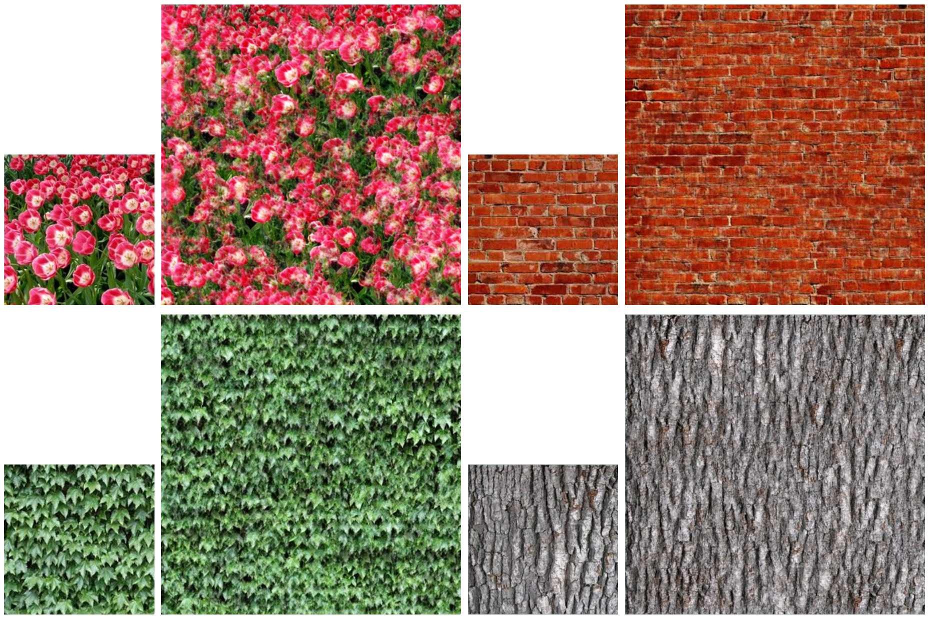


Figure 10.24: Modeling texture patterns. For each example, *Left:* the 224×224 observed image. *Right:* the 448×448 generated image.

should be especially close to the true prior $p(X)$. In other words, at $\hat{W}$, the posteriors $p(X|Y, \hat{W})$ of all the data points $Y \sim P_{\text{data}}$ tend to pave the true prior $p(X)$.

From Rubin's multiple imputation point of view [160] of the EM algorithm, the E-step of EM infers $X_i^{(m)} \sim p(X_i|Y_i, W_t)$ for $m = 1, ..., M$, where M is the number of multiple imputations or multiple guesses of X_i. The multiple guesses account for the uncertainty in inferring X_i from Y_i. The M-step of EM maximizes $Q(W) = \sum_{i=1}^{n} \sum_{m=1}^{M} \log p(Y_i, X_i^{(m)}; W)$ to obtain W_{t+1}. For each data point Y_i, W_{t+1} seeks to reconstruct Y_i by $g(X; W)$ from the inferred latent factors $\{X_i^{(m)}, m = 1, ..., M\}$. In other words, the M-step seeks to map $\{X_i^{(m)}\}$ to Y_i. Pooling over all $i = 1, ..., n$, $\{X_i^{(m)}, \forall i, m\} \sim P_{\text{data}}(X; W_t)$, hence the M-step seeks to map $P_{\text{data}}(X; W_t)$ to the data distribution $P_{\text{data}}(Y)$. Of course the mapping from $\{X_i^{(m)}\}$ to Y_i cannot be exact. In fact, $g(X; W)$ maps $\{X_i^{(m)}\}$ to a d-dimensional patch around the D-dimensional Y_i. The local patches for all $\{Y_i, \forall i\}$ 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 $P_{\text{data}}(Z; W)$ shifts towards $p(X)$, thus $g(X; W)$ maps $p(X)$ to $P_{\text{data}}(Y)$.

Figure 10.25 illustrates the learned D-dimensional manifold. We learn a model where $X = (x_1, x_2, \ldots, x_d)$ has $d = 100$ components from 1000 face images randomly selected from the CelebA dataset [120]. The left panel of Figure 10.25 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 X's of the four corner images using the sphere interpolation [37] and then generating the images by the learned ConvNet.

10.2.5 Extracting appearance and geometry, nonlinear generalization of the AAM model

Active Appearance Models (AAM) [24, 107] 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,

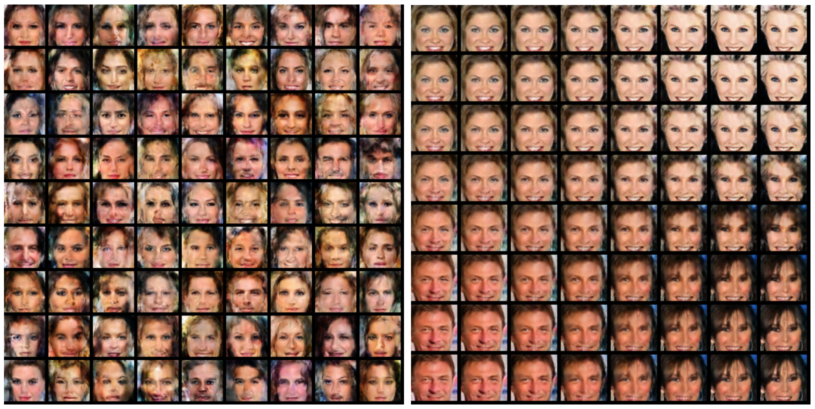


Figure 10.25: Modeling object patterns. *Left:* each image generated by our method is obtained by first sampling $X \sim \mathrm{N}(0, I_{100})$ and then generating the image by $f(X; W)$ with the learned W. *Right:* interpolation. The images at the four corners are reconstructed from the inferred X vectors of four images randomly selected from the training set. Each image in the middle is obtained by first interpolating the X vectors of the four corner images, and then generating the image by $f(X; W)$.

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 [212] 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 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 10.26. 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.

This deformable generator model can be learned by extending the alternating back-propagation algorithm for two latent vectors. After training the model, two sets of basis functions related with the appearance and geometric latent vectors can be learned. Representative basis functions from these two sets are shown in Figure 10.27. The appearance basis function are 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 group 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.

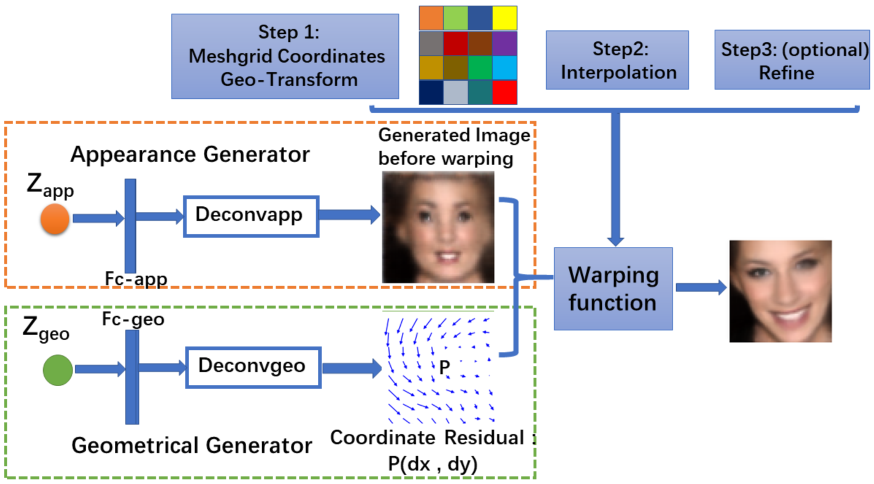


Figure 10.26: An illustration of the proposed model. The model contains two generator networks: one appearance generator and one geometric generator. The two generators are connected with a warping function to produce the final image. The warping function includes a geometric transformation operation for image coordinates and a differentiable interpolation operation. The refining operation is optional for improving the warping function.

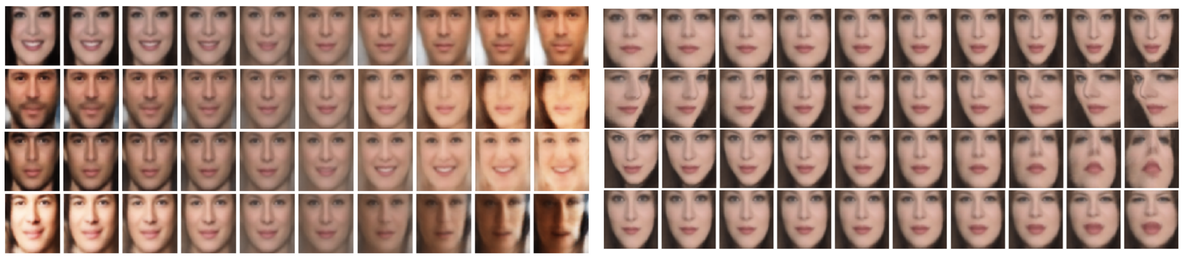


Figure 10.27: Extracting appearance and geometric basis functions. *Left:* Each dimension of the appearance latent vector encodes appearance information such as color, illumination and gender. In the fist line, from left to right, the color of background varies from black to white, and the gender changes from a woman to a man. In the second line, the moustache of the man becomes thicker when the corresponding dimension of Z^a approahces 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 fist 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.

For example, (Figure 10.28 *left*), we can apply the rotation warping and shape warping operations extracted by the deformable generator to the canonical faces to change their shape and pose, while keeping the identity information. For the unseen images, (Figure 10.28 *right*), 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 source image.

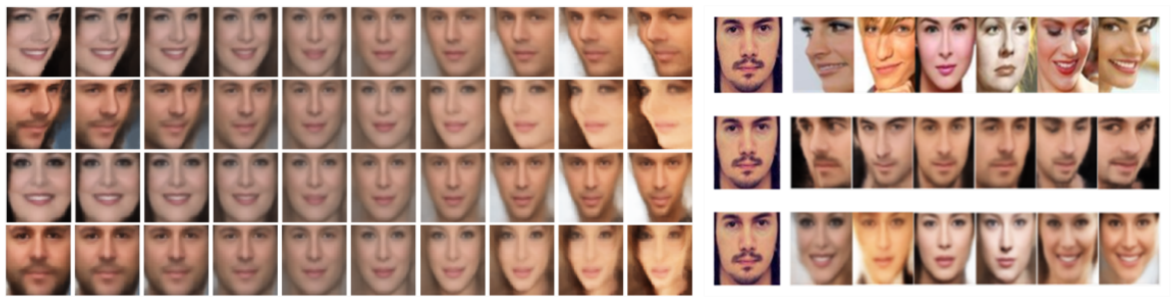


Figure 10.28: Transferring and recombining the abstracted geometric and appearance knowledge. *Left:* Applying the (a) rotation warping and (b) shape warping operations learned by the geometric generator to the canonical faces generated by the appearance generator. Compared with the first two line of the left part of Figure 10.27, only the pose information varies, and the identity information is kept in the process of warping. *Right:* The first row shows the 7 unseen faces from CelebA. The second row shows the generated faces by transferring and recombining the first row's 2th-7th faces' geometric vectors with the first row's 1th face's appearance vector. The third row shows the generated faces by transferring and recombining the first row's 2th-7th faces' appearance vectors with the first row's 1th face's geometric vector.

10.2.6 Dynamic generator model

Let $X = (x_t, t = 1, ..., T)$ be the observed video sequence, where x_t is a frame at time t. The dynamic generator model consists of the following two components:

$$s_t = F_\alpha(s_{t-1}, \xi_t), \tag{10.60}$$
$$x_t = G_\beta(s_t) + \epsilon_t, \tag{10.61}$$

where $t = 1, ..., T$. (10.60) is the transition model, and (10.61) is the emission model. s_t is the d-dimensional hidden state vector. $\xi_t \sim \mathrm{N}(0, I)$ is the noise vector of a certain dimensionality. The Gaussian noise vectors $(\xi_t, t = 1, ..., T)$ are independent of each other. The sequence of $(s_t, t = 1, ..., T)$ follows a non-linear auto-regressive model, where the noise vector ξ_t encodes the randomness in the transition from s_{t-1} to s_t in the d-dimensional state space. F_α is a feedforward neural network or multi-layer perceptron, where α denotes the weight and bias parameters of the network. We can adopt a residual form [76] for F_α 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_β is a top-down convolutional network (sometimes also called deconvolution network), where β denotes the weight and bias parameters of this top-down network. $\epsilon_t \sim \mathrm{N}(0, \sigma^2 I_D)$ is the residual error. We let $\theta = (\alpha, \beta)$ denote all the model parameters.

Let $\xi = (\xi_t, t = 1, ..., T)$. ξ consists of the latent random vectors that need to be inferred from X. Although x_t is generated by the state vector s_t, $S = (s_t, t = 1, ..., T)$ are generated by ξ. In fact, we can write $X = H_\theta(\xi) + \epsilon$, where H_θ composes F_α and G_β over time, and $\epsilon = (\epsilon_t, t = 1, ..., T)$ denotes the observation errors.

Let $p(\xi)$ be the prior distribution of ξ. Let $p_\theta(X|\xi) \sim \mathrm{N}(H_\theta(\xi), \sigma^2 I)$ be the conditional distribution of X given ξ, 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 ξ integrated out. We estimate the model parameter θ 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 $\log p_\theta(\xi, X)$:

$$\begin{aligned} \frac{\partial}{\partial\theta}\log p_\theta(X) &= \frac{1}{p_\theta(X)}\frac{\partial}{\partial\theta}p_\theta(X) \\ &= \frac{1}{p_\theta(X)}\int\left[\frac{\partial}{\partial\theta}\log p_\theta(\xi, X)\right]p_\theta(\xi, X)d\xi \\ &= \mathcal{E}_{p_\theta(\xi|X)}\left[\frac{\partial}{\partial\theta}\log p_\theta(\xi, X)\right], \end{aligned} \tag{10.62}$$

where $p_\theta(\xi|X) = p_\theta(\xi, X)/p_\theta(X)$ is the posterior distribution of the latent ξ 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:

$$\xi^{(\tau+1)} = \xi^{(\tau)} + \frac{\delta^2}{2}\frac{\partial}{\partial\xi}\log p_\theta(\xi^{(\tau)}|X) + \delta z_\tau, \tag{10.63}$$

where τ indexes the time step of the Langevin dynamics (not to be confused with the time step of the dynamics model, t), $z_\tau \sim \mathrm{N}(0, I)$ where I is the identity matrix whose dimension matches that of ξ, and $\xi^{(\tau)} = (\xi_t^{(\tau)}, t = 1, ..., T)$ denotes all the sampled latent noise vectors at time step τ. δ 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 θ by stochastic gradient ascent

$$\Delta\theta \propto \frac{\partial}{\partial\theta}\log p_\theta(\xi, X), \tag{10.64}$$

where the stochasticity of the gradient ascent comes from the fact that we use Monte Carlo to approximate the expectation in (10.62). The learning algorithm iterates the following two steps. (1) Inference step: Given the current θ, sample ξ from $p_\theta(\xi|X)$ according to (11.34). (2) Learning step: Given ξ, update θ according to (10.64). We can use a warm start scheme for sampling in step (1). Specifically, when running the Langevin dynamics, we start from the current ξ, and run a finite number of steps. Then we update θ in step (2) using the sampled ξ. Such a stochastic gradient ascent algorithm has been analyzed by [214].

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 ξ or θ. Step (1) needs to compute the derivative of $\log p_\theta(\xi, X)$ with respect to ξ. Step (2) needs to compute the derivative of $\log p_\theta(\xi, X)$ with respect to θ. 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 $\sigma^2 = 1$)

$$L(\theta, \xi) = -\frac{1}{2}\sum_{t=1}^{T}\left[\|x_t - G_\beta(s_t)\|^2 + \|\xi_t\|^2\right]. \tag{10.65}$$

The derivative with respect to β is

$$\frac{\partial L}{\partial \beta} = \sum_{t=1}^{T}(x_t - G_\beta(s_t))\frac{\partial G_\beta(s_t)}{\partial \beta}. \tag{10.66}$$

The derivative with respect to α is

$$\frac{\partial L}{\partial \alpha} = \sum_{t=1}^{T}(x_t - G_\beta(s_t))\frac{\partial G_\beta(s_t)}{\partial s_t}\frac{\partial s_t}{\partial \alpha}, \tag{10.67}$$

where $\frac{\partial s_t}{\partial \alpha}$ can be computed recursively. To infer ξ, for any fixed time point t_0,

$$\frac{\partial L}{\partial \xi_{t_0}} = \sum_{t=t_0+1}^{T}(x_t - G_\beta(s_t))\frac{\partial G_\beta(s_t)}{\partial s_t}\frac{\partial s_t}{\partial \xi_{t_0}} - \xi_{t_0}, \tag{10.68}$$

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 $s_0 \sim \mathrm{N}(0, I)$. In the inference step, we can sample s_0 together with ξ 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 studied in this paper, it is not a simple task to design an inference model that infers the sequence of latent vectors $\xi = (\xi_t, t = 1, ..., T)$ from the sequence of $X = (x_t, t = 1, ..., T)$. 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 ξ_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 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 [90] 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 [101] 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 (10.60) and (10.61) 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 chunk. In this experiment, the length of 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 10.29 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.

10.2.7 Unsupervised clustering and semi-supervised classification

The generator model can be generalized to incorporate class label y. We can then let $x = g_\theta(y, z) + \epsilon$. For each x, we can sample y and z from their posterior distribution by the Gibbs sampler. This enables us to perform unsupervised clustering. For semi-supervised learning where y is observed for some x, we can simply fix y at the observed value and only sample z in infernece.

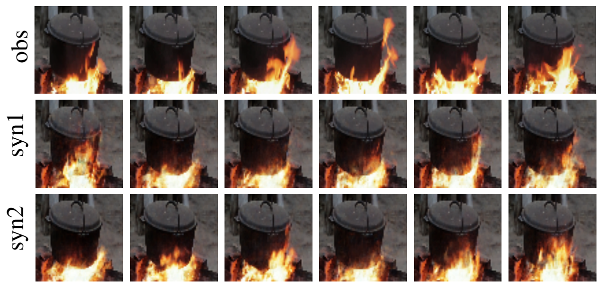
Figure 10.30 shows results on unsupervised clustering. Table 10.1 displays numerical evaluations.

Method	K	MNIST	SVHN	STL-10
K-means	10	53.49%	28.40%	–
AAE [123]	16	83.48%	80.01%	–
DEC [211]	10	84.30%	80.62%	11.90%
VaDE [93]	10	94.46%	84.45%	–
HashGAN [16]	10	96.50%	39.40%	–
CVAE [105]	10	82.26%	62.37%	58.25%
IIC [92]	10	**99.2%**	–	59.6%
Our method	10	98.35%	**85.15%**	**75.30%**

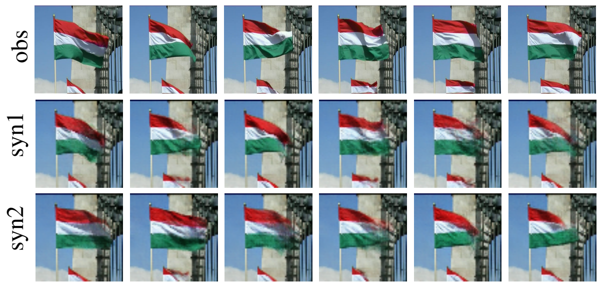
Table 10.1: Comparison of unsupervised clustering accuracy (ACC) for various methods on different datasets.

For semi-supervised learning, we consider three widely adopted benchmark datasets: MNIST, SVHN and CIFAR-10 [108]. Following the standard routine as in previous works, we randomly sample 100, 1,000 and 4,000 labelled images from the MNIST, SVHN and CIFAR-10 respectively during training, and use their testing sets for classification evaluation. We compare our model with the semi-supervised baseline models. Table 10.2 shows the comparison results.

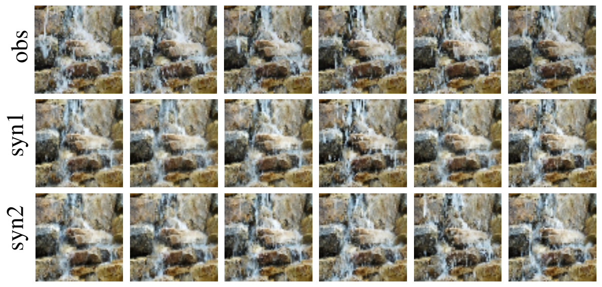
The learned generative model leads to a short-run inference dynamics for classification. We have also tested it on adversarial robustness for classification tasks and obtained state of the art results. The top-down model can easily explain away adversarial perturbations.



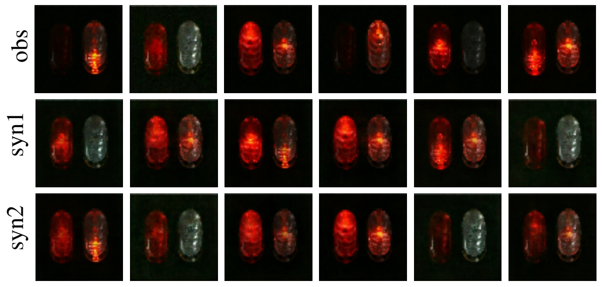
(a) burning fire heating a pot



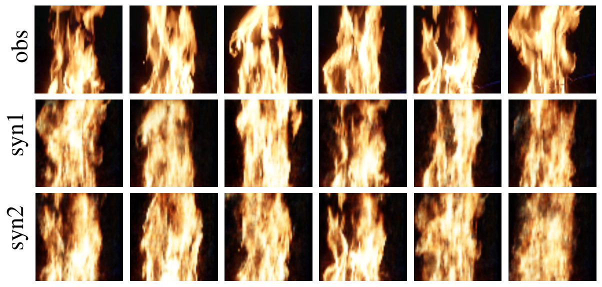
(b) flapping flag



(c) waterfall



(d) flashing lights



(e) flame

Figure 10.29: 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.

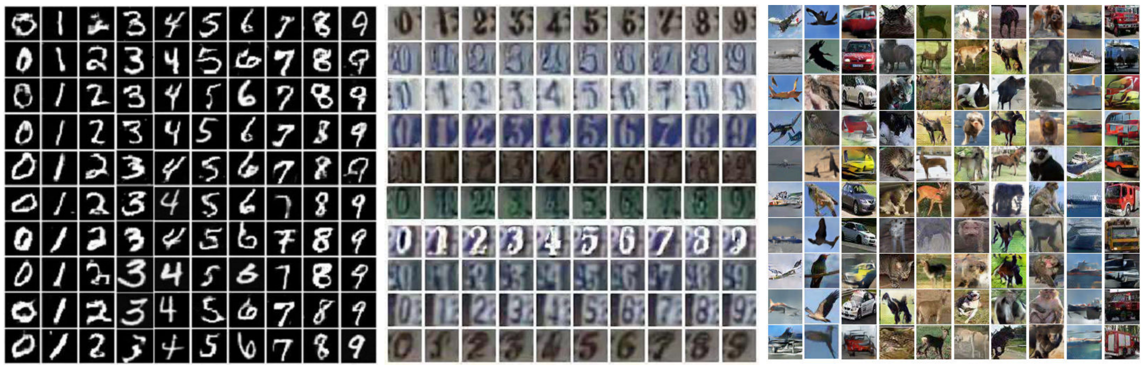


Figure 10.30: Generated samples by our proposed method. Each row shares the same z and each column shares the same y. Left: Generate samples on the MNIST dataset [111]. Middle: Generated samples on the SVHN dataset [139]. Right: Generated samples on the STL-10 dataset [23].

Method	MNIST (n=100)	SVHN (n=1000)	CIFAR-10 (n=4000)
Ladder [151]	1.06	-	20.40
Conv-Ladder [151]	0.89	-	-
CatGAN [176]	1.39	-	19.58
Improved-GAN [164]	0.93	8.11	18.63
VAT [132]	2.33	-	24.63
Triple-GAN [21]	0.91	5.77	16.99
CVAE(M1+M2) [105]	3.33	36.02	-
Our method	**0.89**	**5.62**	**14.25**

Table 10.2: Comparison of semi-supervised classification error rate (%) for various methods on three datasets.

10.2.8 Short-run inference dynamics

We can use a short run inference dynamics guided by the posterior distribution of the latent variables as an approximate inference engine. To make it more concrete, we employ a finite-step gradient descent on the negative log-posterior distribution of the latent variables. For each training example, within each learning iteration, we always initialize such a short run gradient flow from the prior distribution such as Gaussian or uniform noise distribution, and run a finite number (e.g., 20) of steps of gradient descent updates. This amounts to a residual network or a recurrent neural network (RNN) that transforms the initial noise distribution to an approximate posterior distribution. We optimize the step size of the gradient flow by minimizing the Kullback-Leibler divergence between the approximate distribution produced by this short run inference dynamics and the true posterior distribution. Thanks to the computing capacities of the modern deep learning platforms, it is possible to compute the approximate distribution and its entropy, and optimize the aforementioned Kullback-Leibler divergence. This is similar to variational inference, except that the variational parameter is the step size of the gradient descent, or in general, the tuning parameters of the short run inference dynamics. Our experiments show that the proposed method outperforms the VAE in terms of reconstruction error and synthesis quality.

This method is similar to MCMC posterior sampling except that we focus on the optimization of a short run version of MCMC sampling. Such short run inference dynamics is easily affordable on current deep learning platforms and there is no much difficulty to scale it up to big datasets. The method is also similar to variational inference except that the inference model is simply a noise initialized finite-step gradient flow, where the only extra parameter is the step sizes of the gradient flow or in general some tuning parameters of the short run inference dynamics.

One major advantage of the proposed method is that it is natural and automatic. For models with multiple layers of latent variables that may be organized in complex top-down architectures, the gradient descent update of the log-posterior of the latent variables can be automatically obtained on modern deep learning platforms. Such gradient descent update naturally integrates explaining-away competitions and bottom-up and top-down interactions between multiple layers of latent variables. By optimizing the step size of the gradient descent update using the variational criterion, it is possible to have a good approximate posterior distribution and an efficient inference engine. It thus enables researchers to explore sophisticated generative models without worrying about constructing the presumably more sophisticated inference models.

Specifically, we use the following short run inference dynamics for the generator model:

$$z_0 \sim p(z),\ z_{k+1} = z_k + s\frac{\partial}{\partial z}\log p_\theta(z_k|x),\ k = 1, ..., K, \tag{10.69}$$

where $p(z)$ is the prior distribution of z, i.e., $\mathcal{N}(0, I_d)$. Starting from $p(z)$, we run K (e.g., $K = 20$) steps of gradient descent on $-\log p_\theta(z|x)$ with step size s, and we take z_K to be an approximate sample from $p_\theta(z|x)$.

We can write the above dynamics as

$$z_0 \sim p(z),\ z_{k+1} = z_k + sR(z_k), k = 1, ..., K, \tag{10.70}$$

where $R(z) = \frac{\partial}{\partial z}\log p_\theta(z|x)$, where we omit x and θ in $R(z)$ for the simplicity of notation. For finite K, this dynamics is a K-layer residual network, or K-step recurrent neural network.

To further simplify the notation, we may write the dynamics as

$$z_0 \sim p(z),\ z_K = F(z_0), \tag{10.71}$$

where F composes the K steps of gradient descent. The above model can be considered a flow-based model, where F consists of K-step flow of gradient updates. Let the distribution of z_K be $q_s(z)$, where we include the notation s to make it explicit that the distribution of z_K depends on the step size s. Recall that the distribution of z_K also depends on x and θ, so that in full notation, we may write $q_s(z)$ as $q_{s,\theta}(z|x)$.

By change of variable,

$$z_K \sim q_s(z) = p(F^{-1}(z))|\det(dF^{-1}(z)/dz)|. \tag{10.72}$$

We may treat $q_s(z)$ as a flow-like variational inference model.

We want to optimize the step size s so that $q_s(z)$ best approximates the true posterior $p_\theta(z|x)$. This can be accomplished by

$$\min_s \mathrm{KL}(q_s(z)\|p_\theta(z|x)). \tag{10.73}$$

This is similar to variational approximation, with step size s being the variational parameter.

$$\mathrm{KL}(q_s(z)\|p_\theta(z|x)) = \mathrm{E}_{q_s(z)}[\log q_s(z) - \log p_\theta(x, z)] + \log p_\theta(x), \tag{10.74}$$

where the last term $\log p_\theta(x)$ is independent of s, and for the first two terms on the right hand side,

$$\mathrm{E}_{q_s(z)}[\log p_\theta(x, z)] = \mathrm{E}_{p(z_0)}[\log p_\theta(x, F(z_0))], \tag{10.75}$$

$$\mathrm{E}_{q_s(z)}[\log q_s(z)] = \mathrm{E}_{p(z_0)}[\log p(z_0)] - \log|\det(dF(z_0)/dz_0)|]. \tag{10.76}$$

In the above two equations, $\mathrm{E}_{p(z_0)}$ can be approximated by random samples from $z_0 \sim p(z)$, e.g., $\mathcal{N}(0, I_d)$.

In the above computations, we do not need to invert F, but we need to compute the log determinant of the Jacobian $dF(z_0)/dz_0$. Here it is surprisingly fortunate that on modern deep learning platforms with auto-differentiations based on computational graphs, such computation is easily feasible even if the dimension of z_0 is very high. For a sampled $z_0 \sim p(z)$, after computing the matrix $dF(z_0)/dz_0$, we can compute the eigenvalues of $dF(z_0)/dz_0$, so that the log-determinant is the sum of the log of the eigenvalues. Then we can optimize the step size s by minimizing $\mathrm{KL}(q_s(z)\|p_\theta(z|x))$ via a grid search or gradient descent (which is still computable).

Here we assume a constant step size s. We can also allow varying step sizes $s = (s_k, k = 1, ..., K)$, and optimize over (s_k). We may also add momentum to gradient descent dynamics and optimize associated algorithmic parameters.

The learning with short-run inference is based on

$$\mathrm{KL}(p_{\mathrm{data}}(x)q_{s,\theta_t}(z|x)\|p(z)p_\theta(x|z)) = \mathrm{KL}(p_{\mathrm{data}}(x)\|p_\theta(x)) + \mathrm{KL}(q_{s,\theta_t}(z|x)\|p_\theta(z|x)), \tag{10.77}$$

where minimizing $\mathrm{KL}(q_{s,\theta_t}(z|x)\|p_\theta(z|x))$ with respect to s minimizes the perturbation from MLE (a subtlety in notation is that in this conditional KL-divergence, we also average over $p_{\mathrm{data}}(x)$, i.e., we minimize s shared by all the training examples).

Each learning iteration for θ is based on gradient descent of (10.77), which leads to the update

$$\theta_{t+1} = \theta_t + \eta_t \frac{1}{n}\sum_{i=1}^{n} \mathrm{E}_{q_{s,\theta_t}(z_i|x_i)}\left[\frac{\partial}{\partial\theta}\log p_\theta(x_i, z_i) \mid_{\theta=\theta_t}\right], \tag{10.78}$$

where η_t is the step size or learning rate, $\mathrm{E}_{q_{s,\theta_t}(z_i|x_i)}$ (here we use the full notation $q_{s,\theta}(z|x)$ instead of the abbreviated notation $q_s(z)$) can be approximated by sampling from $q_{s,\theta_t}(z_i|x_i)$ using the noise initialized K-step gradient descent. Compared to MLE learning algorithm, we replace $p_{\theta_t}(z|x)$ by $q_{s,\theta_t}(z|x)$, and Monte Carlo samples from $q_{s,\theta}(z|x)$ can be obtained exactly.

The learning algorithm (10.78) solves the following estimating equation:

$$\frac{1}{n}\sum_{i=1}^{n} \mathrm{E}_{q_{s,\theta}(z_i|x_i)}\left[\frac{\partial}{\partial\theta}\log p_\theta(x_i, z_i)\right] = 0, \tag{10.79}$$

which is a perturbation of the MLE estimating equation.

The learning algorithm is a Robbins-Monro algorithm for stochastic approximation to solve the above estimating equation. For fixed s, its convergence follows from regular conditions of Robbins-Monro. Unlike the original maximum likelihood learning, $q_{s,\theta}(z|x)$ can be sampled exactly so that Robbins-Monro theory applies. This greatly simplifies the analysis of the algorithm.

The bias of the learned θ based on the short run inference dynamics relative to the MLE depends on the gap between $q_{s,\theta}(z|x)$ and $p_\theta(z|x)$. Based on the left-hand side of (10.77), while the short-run $q_{s,\theta_t}(z|x)$ seeks to get close to the model by optimizing its s, the model also tends to get close to the short-run inference by updating θ. Thus the bias may actually be beneficial to the short-run inference.

We evaluate the learned generator $g_\theta(z)$ fidelity of generated examples quantitatively on various datasets, each reduced to $40,000$ observed examples. Table 10.3 (a) compares the Fréchet Inception Distance (FID) with Inception v3 classifier [179] on $40,000$ generated examples. Despite its simplicity, short run inference dynamics is competitive to elaborate means of inference in VAE models.

We evaluate the accuracy of the learned inference dynamics $q_{s,\theta_t}(z|x_i)$ by reconstructing test images. In contrast to traditional MCMC posterior sampling with persistent chains, short run inference with small K

	MNIST		SVHN		CelebA	
Models	MSE	FID	MSE	FID	MSE	FID
VAE, L=1	0.020	-	0.019	46.78	0.031	69.90
VAE, L=3	0.018	-	0.015	41.72	0.029	58.33
VAE, L=5	0.018	-	0.014	39.26	0.028	53.40
Ours, L=1	0.019	-	0.018	44.86	0.019	45.80
Ours, L=3	0.017	-	0.015	39.02	0.018	41.20
Ours, L=5	0.015	-	0.011	35.23	0.011	37.29

Table 10.3: Comparison of generators $g_\theta(z)$ with varied number of latent layers L learned by VAE and short run inference with respect to MSE of reconstructions and FID of generated samples of size for MNIST (28×28), SVHN $(32 \times 32 \times 3)$, and CelebA $(32 \times 32 \times 3)$ datasets. (FID on MNIST is not considered meaningful or reflecting human judgment of synthesis quality and hence omitted.)

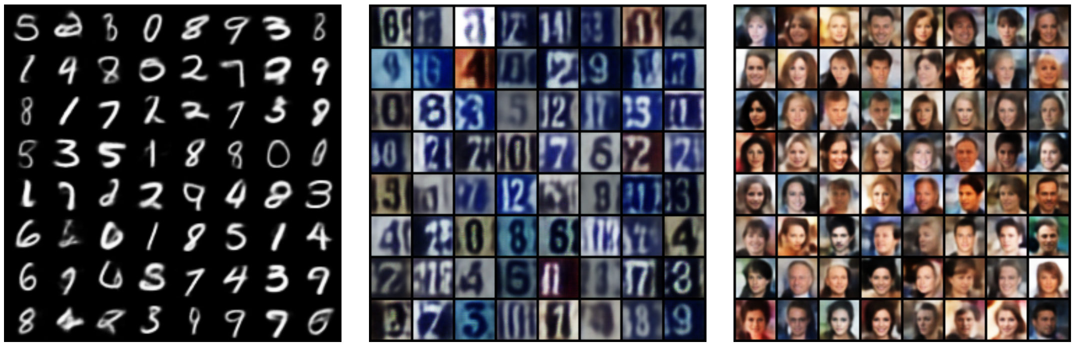


Figure 10.31: Generated samples for $K = 20$ short run inference steps with $L = 5$ layers. From left to right: (1) MNIST (28 $\times$ 28), (2) SVHN (32 $\times$ 32 $\times$ 3), (3) CelebA (32 $\times$ 32 $\times$ 3).

allows not only for efficient learning on training examples, but also the same dynamics can be recruited for testing examples.

Figure 10.32 compares the reconstructions of learned generators with $L = 5$ layers by VAE and short run inference on CelebA $(64 \times 64 \times 3)$. The fidelity of reconstructions by short run inference appears qualitatively improved over VAE, which is quantitatively confirmed by a consistently lower MSE in Table 10.3.

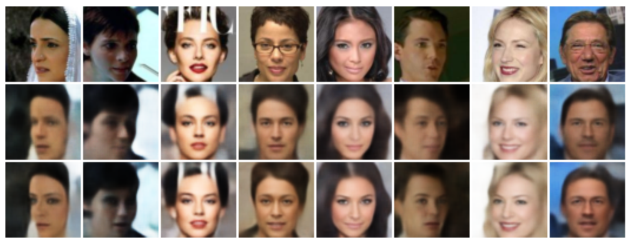


Figure 10.32: Comparison of reconstructions between VAE samples and our method on CelebA ($64\times64\times3$) with $L = 5$. *Top*: original test images. *Middle*: reconstructions from VAE. *Bottom*: reconstructions by short run inference.

11

A Tale of Three Families: Discriminative, Generative and Descriptive Models

11.1 Introduction

In this chapter, we shall review these three families of models within a common framework and explore their connections. We shall start from the flat linear forms of these models. Then we shall present the hierarchical non-linear models, where the non-linear mappings in these models are parametrized by neural networks [109, 111] that have proved exceedingly effective in approximating non-linear relationships.

Currently the most successful family of models are the discriminative models. A discriminative model is in the form of the conditional distribution of the class label given the input signal. The normalizing constant of such a probability model is a summation over the finite number of class labels or categories. It is readily available, so that the model can be easily learned from big datasets. The learning of the descriptive models and the generative models can be much more challenging. A descriptive model is defined as a probability distribution of the signal, which is usually of a high dimensionality. The normalizing constant of such a model is an integral over the high dimensional signal and is analytically intractable. A generative model involves latent variables that follow some prior distribution, so that the marginal distribution of the observed signal is obtained by integrating out the latent variables, and this integral is also analytically intractable. Due to the intractabilities of the integrals in the descriptive and generative models, the learning of such models usually requires Markov chain Monte Carlo (MCMC) sampling [59, 121]. Specifically, the learning of the descriptive models require MCMC sampling of the synthesized signals, while the learning of the generative models require MCMC sampling of the latent variables. Nonetheless, we shall show that such learning methods work reasonably well, where the gradient-based Langevin dynamics [138] can be employed conveniently for MCMC sampling, which is an inner loop within the gradient-based learning of the model parameters.

Because of the high capacity of the neural networks in approximating highly non-linear mappings, the boundary between representation and computation is blurred in neural networks. A deep neural network can be used to represent how the signal is generated or how the features are defined. It can also be used to approximate the solution of a computational problem such as optimization or sampling. For example, the iterative sampling of the latent variables of a generative model can be approximated by an inference model that provides the posterior samples directly, as is the case with the wake-sleep algorithm [79] and the variational auto-encoder (VAE) [102, 133, 154]. As another example, the iterative sampling of a descriptive model can be approximated by a generative model that can generate the signal directly [206]. In general, the solutions to the on-line computational problems can be encoded by high capacity neural networks, so that iterative computations only occur in the off-line learning of the model parameters.

The three families of models do not exist in isolation. There are intimate connections between them. [69, 71] proposed to integrate the descriptive and generative models into a hierarchical model. [186, 187] proposed data-driven MCMC where the MCMC is to fit the generative models, but the proposal distributions for MCMC transitions are provided by discriminative models. The discriminative model and the descriptive model can be translated into each other via the Bayes rule. Tu [185] exploited this relationship to learn the descriptive model via discriminative training, thus unifying the two models. Similarly, the discriminative model can be paired with the generative model in the generative adversarial networks (GAN) [64], and the adversarial learning has become an alternative framework to likelihood-based learning. The descriptive model and the generative model can also be paired up so that they can jump-start each other's MCMC sampling. Moreover, the family of descriptive models and the family of generative models overlap in terms of undirected latent energy-based models.

11.2 Non-hierarchical linear forms of the three families

We shall first review the non-hierarchical linear forms of the there families of models within a common framework.

11.2.1 Discriminative models

This subsection reviews the linear form of the discriminative models.

The table below displays the dataset for training the discriminative models.

	input	features	output
1	$X_1^\top$	$h_1^\top$	Y_1
2	$X_2^\top$	$h_2^\top$	Y_2
...			
n	$X_n^\top$	$h_n^\top$	Y_n

There are n training examples. For the i-th example, let $X_i = (X_{ij}, j = 1, ..., p)^\top$ be the p-dimensional input signal (the (n, p) notation is commonly used in statistics to denote the number of observations and the number of predictors). Let Y_i be the outcome label. In the case of classification, Y_i is categorical or binary. $h_i = (h_{ik}, k = 1, ..., d)^\top$ is the d-dimensional vector of features or hidden variables.

The discriminative models can be represented by the diagram below,

$$\begin{array}{rl} \text{output}: & Y_i \\ & \uparrow \\ \text{features}: & h_i \\ & \uparrow \\ \text{input}: & X_i \end{array} \tag{11.1}$$

where the vector of features h_i is computed from X_i via $h_i = h(X_i)$. In a non-hierarchical or flat model, the feature vector h_i is designed, not learned, i.e., $h()$ is a pre-specified non-linear transformation.

For the case of binary classification where $Y_i \in \{+1, -1\}$, (Y_i, X_i) follow a logistic regression

$$\log \frac{\Pr(Y_i = +|X_i)}{\Pr(Y_i = -|X_i)} = h_i^\top \theta + b, \tag{11.2}$$

where θ is the d-dimensional vector of weight or coefficient parameters, and b is the bias or intercept parameter. The classification can also be based on the perceptron model

$$\hat{Y}_i = \text{sign}(h_i^\top \theta + b), \tag{11.3}$$

where $\text{sign}(r) = +1$ if $r \geq 0$, and $\text{sign}(r) = -1$ otherwise. Both the logistic regression and the perceptron can be generalized to the multi-category case. The bias term b can be absorbed into the weight parameters θ if we fix $h_{i1} = 1$.

Let $f(X) = h(X)^\top \theta$. $f(X)$ captures the relationship between X and Y. Because $h(X)$ is non-linear, $f(X)$ is also non-linear. We say the model is in the linear form because it is linear in θ, or $f(X)$ is a linear combination of the features in $h(X)$. The following are the choices of $h()$ in various discriminative models.

Kernel machine [25]: $h_i = h(X_i)$ is implicit, and the dimension of h_i can potentially be infinite. The implementation of this method is based on the kernel trick $\langle h(X), h(X') \rangle = K(X, X')$, where K is a kernel that is explicitly used by the classifier such as the support vector machine [25]. $f(X) = h(X)^\top \theta$ belongs to the reproducing kernel Hilbert space where the norm of f can be defined as the Euclidean norm of θ, and the norm is used to regularize the model. A Bayesian treatment leads to the Gaussian process, where θ is assumed to follow $\text{N}(0, \sigma^2 I_d)$, and I_d is the identity matrix of dimension d. $f(X)$ is a Gaussian process with $\text{Cov}(f(X), f(X')) = \sigma^2 K(X, X')$.

Boosting machine [48]: For $h_i = (h_{ik}, k = 1, ..., d)^\top$, each $h_{ik} \in \{+, -\}$ is a weak classifier or a binary feature extracted from X, and $f(X) = h(X)^\top \theta$ is a committee of weak classifiers.

CART [14]: In the classification and regression trees, there are d rectangle regions $\{R_k, k = 1, ..., d\}$ resulted from recursive binary partition of the space of X, and each $h_{ik} = 1(X_i \in R_k)$ is the binary indicator such that $h_{ik} = 1$ if $X_i \in R_k$ and $h_{ik} = 0$ otherwise. $f(X) = h(X)^\top \theta$ is a piecewise constant function.

MARS [53]: In the multivariate adaptive regression splines, the components of $h(X)$ are hinge functions such as $\max(0, x_j - t)$ (where x_j is the j-th component of X, $j = 1, ..., p$, and t is a threshold) and their products. It can be considered a continuous version of CART.

Encoder and decoder: In the diagram in (11.1), the transformation $X_i \to h_i$ is called an encoder, and the transformation $h_i \to Y_i$ is called a decoder. In the non-hierarchical model, the encoder is designed, and only the decoder is learned. This is different from the auto-encoder in unsupervised learning.

The outcome Y_i can also be continuous or a high-dimensional vector. The learning then becomes a regression problem. Both classification and regression are about supervised learning because for each input X_i, an output Y_i is provided as supervision. The reinforcement learning is similar to supervised learning except that the guidance is in the form of a reward function.

11.2.2 Descriptive model

This subsection describes the linear form of the descriptive models and the maximum likelihood learning algorithm.

The descriptive models [220] can be learned in the unsupervised setting, where Y_i are not observed, as illustrated by the table below.

	input	features	output
1	$X_1^\top$	$h_1^\top$	?
2	$X_2^\top$	$h_2^\top$	?
...			
n	$X_n^\top$	$h_n^\top$	?

The linear form of the descriptive model is an exponential family model. It specifies a probability distribution on the signal X_i via an energy function that is a linear combination of the features,

$$p_\theta(X) = \frac{1}{Z(\theta)} \exp\left[h(X)^\top \theta\right] p_0(X) \tag{11.4}$$

where $h(X)$ is the d-dimensional feature vector extracted from X, and θ is the d-dimensional vector of weight parameters. $p_0(X)$ is a known reference distribution such as the white noise model $X \sim \mathrm{N}(0, \sigma^2 I_p)$, or the uniform distribution within a bounded range.

$$Z(\theta) = \int \exp[h(X)^\top \theta] p_0(X) dX = \mathrm{E}_{p_0}\{\exp[h(X)^\top \theta]\} \tag{11.5}$$

is the normalizing constant (E_p denotes the expectation with respect to p). It is analytically intractable.

The descriptive model (11.4) has the following information theoretical property [3, 31, 224]. Let P_{data} be the distribution that generates the training data $\{X_i\}$. Let $\Theta = \{p_\theta, \forall \theta\}$ be the family of distributions defined by the descriptive model. Let $\Omega = \{p : \mathrm{E}_p[h(X)] = \hat{h}\}$, where $\hat{h} = \mathrm{E}_{P_{\text{data}}}[h(X)]$. $\hat{h}$ can be estimated from the observed data by the sample average: $\sum_{i=1}^n h(X_i)/n$. Ω is the family of distributions that reproduce the observed $\hat{h}$. Let $\hat{p} = p_{\hat{\theta}} \in \Theta \cup \Omega$ be the intersection between Θ and Ω. Then for

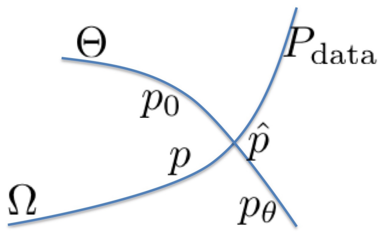


Figure 11.1: The two curves illustrate Θ and Ω respectively, where each point is a probability distribution.

any $p_\theta \in \Theta$ and any $p \in \Omega$, we have $\mathrm{KL}(p||p_\theta) = \mathrm{KL}(p||\hat{p}) + \mathrm{KL}(\hat{p}||p_\theta)$, which can be interpreted as a Pythagorean property that defines orthogonality. $\mathrm{KL}(p||q) = \mathrm{E}_p[\log(p(X)/q(X))]$ denotes the Kullback-Leibler divergence from p to q. Thus Θ and Ω are orthogonal to each other, $\Theta \perp \Omega$, as illustrated by Figure 11.1.

This leads to the following dual properties of $\hat{p}$, which can be considered the learned model:

(1) Maximum likelihood. $\hat{p} = \arg\min_\Theta \mathrm{KL}(P_{\text{data}}||p_\theta)$. That is, $\hat{p}$ is the projection of P_{data} on Θ. $\mathrm{KL}(P_{\text{data}}||p_\theta) = \mathrm{E}_{P_{\text{data}}}[\log P_{\text{data}}(X)] - \mathrm{E}_{P_{\text{data}}}[\log p_\theta(X)]$. The second term $\mathrm{E}_{P_{\text{data}}}[\log p_\theta(X)]$ is the population version of the log-likelihood. Thus minimizing $\mathrm{KL}(P_{\text{data}}|p_\theta)$ is equivalent to maximizing the likelihood.

(2) Maximum entropy: $\hat{p} = \arg\min_\Omega \mathrm{KL}(p||p_0)$. That is, $\hat{p}$ is the minimal modification of p_0 to reproduce the observed feature statistics $\hat{h}$. $\mathrm{KL}(p||p_0) = \mathrm{E}_p[\log p(X)] - \mathrm{E}_p[\log p_0(X)]$. If p_0 is the uniform distribution, then the second term is a constant, and the first term is the negative entropy. In that case, minimizing $\mathrm{KL}(p||p_0)$ is equivalent to maximizing the entropy over Ω.

Given the training data $\{X_i\}$, let $\mathcal{L}(\theta) = \sum_{i=1}^n \log p_\theta(X_i)/n$ be the log-likelihood. The gradient of $\mathcal{L}(\theta)$ is

$$\mathcal{L}'(\theta) = \frac{1}{n} \sum_{i=1}^n h(X_i) - \mathrm{E}_\theta[h(X)], \tag{11.6}$$

because $\partial \log Z(\theta)/\partial\theta = \text{E}_\theta[h(X)]$, where E_θ denotes the expectation with respect to p_θ. This leads to a stochastic gradient ascent algorithm for maximizing $\mathcal{L}(\theta)$,

$$\theta_{t+1} = \theta_t + \eta_t \left[\frac{1}{n}\sum_{i=1}^{n} h(X_i) - \frac{1}{\tilde{n}}\sum_{i=1}^{\tilde{n}} h(\tilde{X}_i) \right], \tag{11.7}$$

where $\{\tilde{X}_i, i = 1, ..., \tilde{n}\}$ are random samples from p_{θ_t}, and η_t is the learning rate. The learning algorithm has an "analysis by synthesis" interpretation. The $\{\tilde{X}_i\}$ are the synthesized data generated by the current model. The learning algorithm updates the parameters in order to make the synthesized data similar to the observed data in terms of the feature statistics. At the maximum likelihood estimate $\hat{\theta}$, the model matches the data: $\text{E}_{\hat{\theta}}[h(X)] = \text{E}_{P_{\text{data}}}[h(X)]$.

One important class of descriptive models are the Markov random field models [13, 60], such as the Ising model in statistical physics. Such models play an important role in the pattern theory.

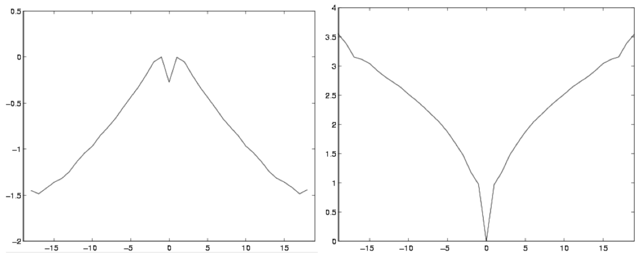


Figure 11.2: Two types of potential functions learned by [221] from natural images. The function on the left encourages big filter responses and creates patterns via reaction, while the function on the right prefers small filter responses and smoothes the synthesized image via diffusion.

One example of the descriptive model (11.4) is the FRAME (Filters, Random field, And Maximum Entropy) model [203, 224], where $h(X)$ consists of histograms of responses from a bank of filters. In a simplified non-convolutional version, $h(X)^\top\theta = f(WX) = \sum_{k=1}^{d} f_k(W_k X)$, where W is a $d \times p$ matrix, and W_k is the k-th row of W. WX consists of the d filter responses with each row of W being a linear filter. $(f_k, k = 1, ..., d)$ are k one-dimensional potential functions applied respectively to the d elements of WX. In the FRAME model, the rows of W are a bank of Gabor wavelets or filters [30]. Given the filters, [221] learned the potential functions $(-f_k, k = 1, ..., d)$ from natural images. There are two types of potential functions as shown in Figure 11.2 taken from [221]. The function on the left encourages big filter responses while the function on the right prefers small filter responses. [221] 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 11.2 is for reaction to create patterns, while the function on the right is for diffusion to smooth out the synthesized image.

In [118], the authors illustrate the idea of learning $W = (W_k, k = 1, ..., d)$ by a two-dimensional example. Each step of the learning algorithm adds a row W_k to the current W. Each row corresponds to a projection of X. Each step finds a direction of the projection that reveals the maximum difference between the data points sampled from the current model and the observed data points. The learning algorithm then updates the model to match the marginal distributions of the model and the data in that direction. After a few steps, the distribution of the learned model is almost the same as the distribution of the observed data. By assuming a parametric differentiable form for $f_k()$, W can be learned by gradient descent. Such models are called product of experts [78, 180] or field of experts [158].

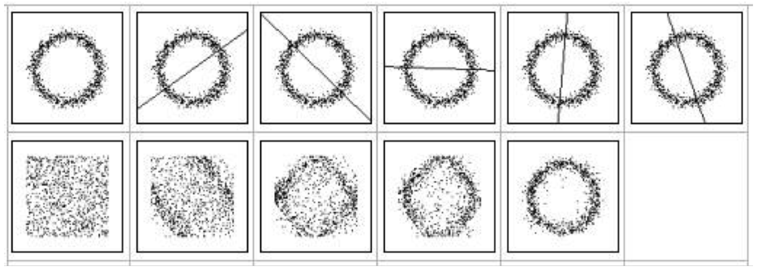


Figure 11.3: Learning a two dimensional FRAME model by sequentially adding rows to W. Each row of W corresponds to a projection of the data. Each step finds the projection that reveals the maximum difference between the observed data and the synthesized data generated by the current model.

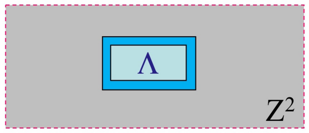


Figure 11.4: Under the uniform distribution of images defined on a large lattice (that goes to Z^2) where the images share the same marginal histograms of filter responses, the conditional distribution of the local image patch given its boundary (in blue color) follows the FRAME model.

The FRAME model is convolutional, where the rows of W can be partitioned into different groups, and the rows in the same group are spatially translated versions of each other, like wavelets. They are called filters or kernels. The model can be justified by a uniform distribution over the images defined on a large lattice that goes to Z^2, where all the images share the same marginal histograms of filter responses. Under such a uniform distribution, the distribution of the local image patch defined on a local lattice Λ conditional on its boundary (illustrated by the blue color, including all the pixels outside Λ that can be covered by the same filters as the pixels within Λ) follows the FRAME model [203]. See Figure 11.4 for an illustration.

11.2.3 Generative models

This subsection reviews various versions of the linear generative models. These models share the same linear form, but they differ in terms of the prior assumptions of the latent factors or coefficients.

Like the descriptive models, the generative models can be learned in the unsupervised setting, where Y_i are not observed, as illustrated below:

	input	hidden	output
1	$X_1^\top$	$h_1^\top$	?
2	$X_2^\top$	$h_2^\top$	?
...			
n	$X_n^\top$	$h_n^\top$	?

In a generative model, the vector h_i is not a vector of features extracted from the signal X_i. h_i is a vector of hidden variables that is used to generate X_i, as illustrated by the following diagram:

$$\begin{array}{rc} \text{hidden}: & h_i \\ & \downarrow \\ \text{input}: & X_i \end{array} \tag{11.8}$$

The components of the d-dimensional h_i are variably called factors, sources, components or causes.

Auto-encoder: h_i is also called a code in the auto-encoder illustrated by the following diagram:

$$\begin{array}{rc} \text{code}: & h_i \\ & \uparrow\downarrow \\ \text{input}: & X_i \end{array} \tag{11.9}$$

The direction from h_i to X_i is called the decoder, and the direction from X_i to h_i is called the encoder. The decoder corresponds to the generative model in (11.8), while the encoder can be considered the inference model.

Distributed representation and disentanglement: $h_i = (h_{ik}, k = 1, ..., d)$ is called a distributed representation of X_i. Usually the components of h_i, $(h_{ik}, k = 1, ..., d)$, are assumed to be independent, and (h_{ik}) are said to disentangle the variations in X_i.

Embedding: h_i can also be considered the coordinates of X_i, if we embed X_i into a low-dimensional space, as illustrated by the following diagram:

$$\begin{array}{c} \leftarrow h_i \rightarrow \\ | \\ \leftarrow X_i \rightarrow \end{array} \tag{11.10}$$

In the training data, we find a h_i for each X_i, so that $\{h_i, i = 1, ..., n\}$ preserve the relative relations between $\{X_i, i = 1, ..., n\}$. The prototype example of embedding is multi-dimensional scaling, where we want to preserve the Euclidean distances between the examples. A more recent example of embedding is local linear embedding [159]. In the embedding framework, there are no explicit encoder and decoder.

Linear generative model: The linear form of the generative model is as follows:

$$X_i = Wh_i + \epsilon_i, \tag{11.11}$$

for $i = 1, ..., n$, where W is a $p \times d$ dimensional matrix (p is the dimensionality of X_i and d is the dimensionality of h_i), and ϵ_i is a p-dimensional residual vector. The following are the interpretations of W:

(1) Loading matrix: Let $W = (w_{jk})_{p\times d}$. $x_{ij} \approx \sum_{k=1}^{d} w_{jk}h_{ik}$, i.e., each component of X_i, x_{ij}, is a linear combination of the latent factors. w_{jk} is the loading weight of factor k on variable j.

(2) Basis vectors: Let $W = (W_k, k = 1, ..., d)$, where W_k is the k-th column of W. $X_i \approx \sum_{k=1}^{d} h_{ik}W_k$, i.e., X_i is a linear superposition of the basis vectors (W_k), where h_{ik} are the coefficients.

(3) Matrix factorization: $(X_1, ..., X_n) \approx W(h_1, ..., h_n)$, where the $p \times n$ matrix $(X_1, ..., X_n)$ is factorized into the $p \times d$ matrix W and the $d \times n$ matrix $(h_1, ..., h_n)$.

The following are some of the commonly assumed prior distributions or constraints on h_i.

Factor analysis [161]: $h_i \sim \mathrm{N}(0, I_d)$, $X_i = Wh_i + \epsilon_i$, $\epsilon_i \sim \mathrm{N}(0, \sigma^2 I_p)$, and ϵ_i is independent of h_i. The dimensionality of h_i, which is d, is smaller than the dimensionality of X_i, which is p. The factor analysis is very similar to the principal component analysis (PCA), which is a popular tool for dimension reduction. The difference is that in factor analysis, the column vectors of W do not need to be orthogonal to each other.

The factor analysis model originated from psychology, where X_i consists of the test scores of student i on p subjects. h_i consists of the verbal intelligence and the analytical intelligence of student i ($d = 2$).

Another example is the decathlon competition, where X_i consists of the scores of athlete i on $p = 10$ sports, and h_i consists of athlete i's speed, strength and endurance ($d = 3$).

Independent component analysis [89]: In ICA, for $h_i = (h_{ik}, k = 1, ..., d)$, $h_{ik} \sim P_k$ independently, and P_k are assumed to be heavy-tailed distributions. For analytical tractability, ICA assumes that $d = p$, and $\epsilon_i = 0$. Hence $X_i = Wh_i$, where W is a squared matrix assumed to be invertible. $h_i = AX_i$, where $A = W^{-1}$. Let $P(h) = \prod_{k=1}^{d} P_k(h_{ik})$. The marginal distribution of X_i has a closed form $X_i \sim P(AX)|\det(A)|$. The ICA model is both a generative model and a descriptive model.

Sparse coding [144]: In the sparse coding model, the dimensionality of h_i, which is d, is bigger than the dimensionality of X_i, which is p. However, $h_i = (h_{ik}, k = 1, ..., d)$ is a sparse vector, meaning that only a small number of h_{ik} are non-zero, although for different example i, the non-zero elements in h_i can be different. Thus unlike PCA, sparse coding provides adaptive dimension reduction. $W = (W_k, k = 1, ..., d)$ is called a redundant dictionary because $d > p$, and each W_k is a basis vector or a "word" in the dictionary. Each $X_i \approx Wh_i = \sum_{k=1}^{d} h_{ik} W_k$ is explained by a small number of W_k selected from the dictionary, depending on which h_{ik} are non-zero. The inference of the sparse vector h_i can be accomplished by Lasso or basis pursuit [19,181] that minimizes $\sum_{i=1}^{n} \left[\|X_i - Wh_i\|^2 + \lambda |h_i|_{\ell_1} \right]$, which imposes the sparsity inducing ℓ_1 regularization on h_i with a regularization parameter λ.

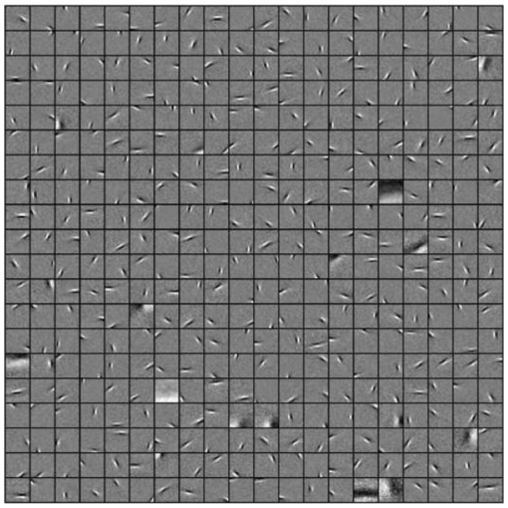


Figure 11.5: Sparse coding: learned basis vectors from natural image patches. Each image patch in the picture is a column vector of W.

A Bayesian probabilistic formulation is to assume a spike-slab prior: $h_{ik} \sim \rho\delta_0 + (1 - \rho)\mathrm{N}(0, \tau^2)$ with a small $1 - \rho$, which is the probability that h_{ik} is non-zero.

Figure 11.5 displays a sparse code learned from a training set of natural image patches of size 12×12 [144]. Each column of W, W_k, is a basis vector that can be made into an image patch as shown in the figure.

Non-negative matrix factorization [114]: In NMF, h_i is constrained to have non-negative components, i.e., $h_{ik} \geq 0$ for all k. It is also called positive factor analysis [146]. The rationale for NMF is that the parts of a pattern should be additive and the parts should contribute positively.

Matrix factorization for recommender system [106]: In recommender system, $X_i = (x_{ij}, j = 1, ..., p)$ are the ratings of user i on the p items. For instance, in the Netflix example, there are n users and p movies, and x_{ij} is user i's rating of movie j. Let $\mathbf{w}_j$ be the j-th row of matrix W, then $x_{ij} = \langle \mathbf{w}_j, h_i \rangle + \epsilon_{ij}$, where h_i characterizes the desires of user i in d aspects, and $\mathbf{w}_j$ characterizes the desirabilities of item j in the corresponding aspects. The rating matrix $(X_i, i = 1, ..., n)$ thus admits a rank d factorization. The rating matrix is in general incomplete. However, we can still estimate (h_i) and $(\mathbf{w}_j)$ from the observed ratings and use them to complete the rating matrix for the purpose of recommendation.

Probabilistic formulation: In the above models, there is a prior model $h_i \sim p(h)$ or a prior constraint such as h_i is sparse or non-negative. Then there is a linear generative model $X_i = Wh_i + \epsilon_i$, with $\epsilon_i \sim \text{N}(0, \sigma^2 I_p)$, for $i = 1, ..., n$. This defines the conditional distribution $p(X|h; W)$. The joint distribution is $p(h)p(X|h; W) = p(h, X|W)$. The marginal distribution is obtained by integrating out h:

$$p(X|W) = \int p(h)p(X|h; W)dh = \int p(h, X|W)dh. \tag{11.12}$$

This integral is analytically intractable. According to the Bayes rule, h can be inferred from X based on the posterior distribution, $p(h|X; W) = p(h, X|W)/p(X|W)$, which is proportional to $p(h, X|W)$ as a function of h. We call $p(h|X; W)$ the inference model.

In the auto-encoder terminology, $p(h)$ and $p(X|h; W)$ define the decoder, while $p(h|X; W)$ defines the encoder. In factor analysis and independent component analysis, h can be inferred in closed form. For other models, however, h needs to be inferred by an iterative algorithm.

Restricted Boltzmann machine [80]: In RBM, unlike the above models, there is no explicit prior $p(h)$. The model is defined by the joint distribution

$$(h_i, X_i) \sim p(h, X|W) = \frac{1}{Z(W)} \exp\left[\sum_{j,k} w_{jk} x_j h_k\right] \tag{11.13}$$

$$= \frac{1}{Z(W)} \exp\left[X^\top W h\right]. \tag{11.14}$$

The above model assumes that both h_i and X_i are binary. Under the above model, both the generative distribution $p(X|h; W)$ and the inference distribution $p(h|X; W)$ are independent logistic regressions. We may modify the model slightly to make X continuous, so that in the modified model, the generative distribution $p(X|h; W)$ is normal linear regression: $X = Wh + \epsilon$, with $\epsilon \sim \text{N}(0, \sigma^2 I_p)$. The inference model, $p(h|X; W)$, is logistic regression, $h \sim \text{logistic}(W^\top X)$, i.e., $\Pr(h_k = 1|X; W) = \text{sigmoid}(\sum_{j=1}^p w_{jk} x_j)$, where $\text{sigmoid}(r) = 1/(1 + e^{-r})$.

If we sum out h, the marginal distribution $p(X|W) = \sum_h p(h, X|W)$ can be obtained in closed form, and $p(X|W)$ is a descriptive model.

RBM-like auto-encoder [11, 191]: The RBM leads to the following auto-encoder: the encoder is $h_k = \text{sigmoid}(\sum_{j=1}^p w_{jk} x_j)$, i.e., $h = \text{sigmoid}(W^\top X)$; the decoder is $X = Wh$.

Like the descriptive model, the generative model can also be learned by maximum likelihood. However, unlike the "analysis by synthesis" scheme for learning the descriptive model, the learning algorithm for generative model follows an "analysis by inference" scheme. Within each iteration of the learning algorithm, there is an inner loop for inferring h_i for each X_i. The most rigorous inference method is to sample h_i from the posterior distribution or the inference distribution $p(h_i|X_i; W)$. After inferring h_i for each X_i, we can then update the model parameters by analyzing the "imputed" dataset $\{(h_i; X_i)\}$, by fitting the generative distribution $p(X|h; W)$. The EM algorithm [32] is an example of this learning scheme, where the inference step is to compute expectation with respect to $p(h_i|X_i; W)$. From a Monte Carlo perspective, it means we make multiple imputations [160] or make multiple guesses of h_i to account for the uncertainties in $p(h_i|X_i; W)$. Then we analyze the multiply imputed dataset to update the model parameters.

11.3 Interactions between different families

11.3.1 Discriminative learning of descriptive model

This subsection shows that the descriptive model can be learned discriminatively.

The descriptive model (11.4) can be connected to the discriminative model (11.2) if we treat $p_0(X)$ as the distribution of the negative examples, and $p_\theta(X)$ as the distribution of the positive examples. Suppose we generate the data as follows: $Y_i \sim \text{Bernoulli}(\rho)$, i.e., $\Pr(Y_i = 1) = \rho$, which is the prior probability of positive examples. $[X_i \mid Y_i = 1] \sim p_\theta(X)$, and $[X_i \mid Y_i = 0] \sim p_0(X)$. According to the Bayes rule

$$\log \frac{\Pr(Y_i = 1 \mid X_i)}{\Pr(Y_i = 0 \mid X_i)} = h(X_i)^\top \theta - \log Z(\theta) + \log[\rho/(1-\rho)], \tag{11.15}$$

which corresponds to (11.2) with $b = -\log Z(\theta) + \log[\rho/(1-\rho)]$.

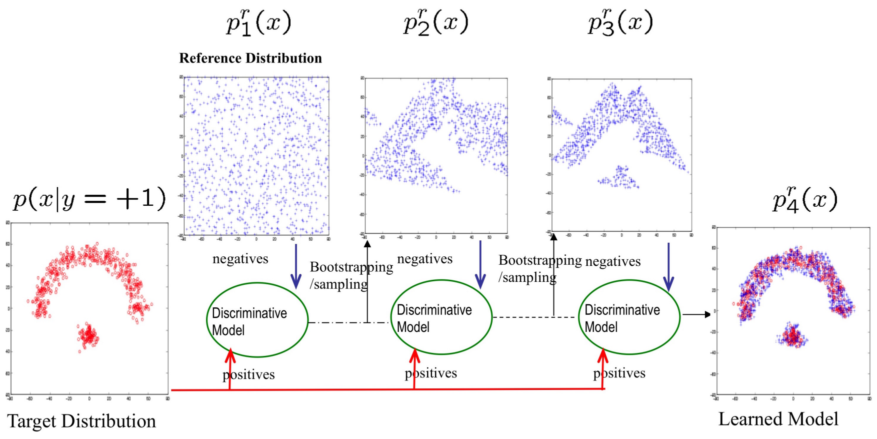


Figure 11.6: Discriminative learning of the descriptive model. By fitting a logistic regression to discriminate between the observed examples and the synthesized examples generated by the current model, we can modify the current model according to the fitted logistic regression, so that the modified model gets closer to the distribution of the observed data.

Tu [185] made use of this fact to estimate p_θ discriminatively. The learning algorithm starts from p_0. At step t, we let the current p_t serve as the negative distribution, and generate synthesized examples from p_t. Then we fit a logistic regression by treating the examples generated by p_t as the negative examples, and the observed examples as the positive examples. Let θ be the estimated parameter of this logistic regression. We then let $p_{t+1}(X) = \exp(h(X)^\top \theta) p_t(X)/Z$. See [185] for an analysis of the convergence of the learning algorithm.

Figure 11.6 taken from [185] illustrates the learning process by starting from the uniform p_0. By iteratively fitting the logistic regression and modifying the distribution, the learned distribution converges to the true distribution.

11.3.2 DDMCMC: integration of discriminative and generative models

In [186, 187] the authors proposed a data-driven MCMC method for fitting the generative models as well as the descriptive models to the data. Fitting such models usually require time-consuming MCMC. In [186,187]

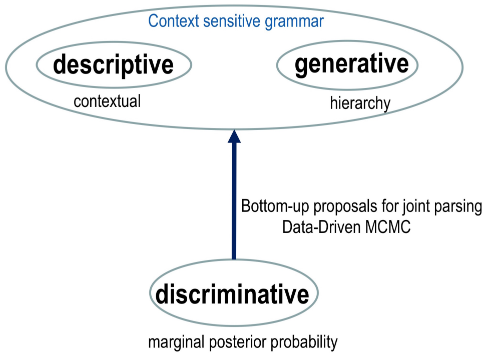

Context sensitive grammar
descriptive
contextual
generative
hierarchy
Bottom-up proposals for joint parsing
Data-Driven MCMC
discriminative
marginal posterior probability

Figure 11.7: Data-driven MCMC: when fitting the generative models and descriptive models using MCMC, the discriminative models can be employed to provide proposals for MCMC transitions.

the authors proposed to speed up the MCMC by using the discriminative models to provide the proposals for the Metropolis-Hastings algorithm. See Figure 11.7 for an illustration.

11.4 Hierarchical forms of the three families

This section presents the hierarchical non-linear forms of the three families of models, where the non-linear mappings are parametrized by neural networks, in particular, the convolutional neural networks.

11.4.1 Recent developments

During the past few years, deep convolutional neural networks (CNNs or ConvNets) [109,111] and recurrent neural networks (RNNs) [81] have transformed the fields of computer vision, speech recognition, natural language processing, and other fields in artificial intelligence (AI). Even though these neural networks were invented decades ago, their potentials were realized only recently mainly because of the following two factors. (1) The availability of big training datasets such as Imagenet [33]. (2) The improvement in computing power, mainly brought by the graphical processing units (GPUs). These two factors, together with some recent clever tweaks and inventions such as rectified linear units [109], batch normalization [90], residual networks [76], etc., enable the training of very deep networks (e.g., 152 layers with 60 million parameters in a residual network for object recognition [76]) that achieve impressive performances on many tasks in AI (a recent example being Alpha Go Zero [171]).

One key reason for the successes of deep neural networks is that they are universal and flexible function approximators. For instance, a feedforward neural network with rectified linear units is a piecewise linear function with recursively partitioned linear pieces that can approximate any continuous non-linear mapping [134]. However, this does not fully explain the "unreasonable effectiveness" of deep neural networks. The stochastic gradient descent algorithm that is commonly employed to train the neural networks is expected to approach only a local minimum of the highly non-convex objective function. However, for large and deep networks, it appears that most of the local modes are equally good [22] in terms of training and testing errors, and the apparent vices of local modes and stochasticity in the mini-batch on-line training algorithm actually turn out to be big virtues in that they seem to prevent overfitting and lead to good generalization [22].

The approximation capacities of the deep neural networks have been extensively exploited in supervised learning (such as classification networks and regression networks) and reinforcement learning (such as policy networks and value networks). They have also proven to be useful for unsupervised learning and generative modeling, where the goal is to learn features or hidden variables from the observed signals without external guidance such as class labels or rewards. The unsupervised learning is often accomplished in the context of a generative model (or an auto-encoder), which explains or characterizes the observed examples.

11.4.2 Discriminative models by convolutional neural networks

The neural networks in general and the convolutional neural networks (ConvNet or CNN) in particular were initially designed for discriminative models. Let X be the p-dimensional input vector, and Y be the output. We want to predict Y by $\hat{Y}$ which is a non-linear transformation of X: $\hat{Y} = f_\theta(X)$, where f is parametrized by parameters θ. In a feedforward neural network, f is a composition of L layers of liner mappings followed by element-wise non-linear rectifications, as illustrated by the following diagram:

$$X \to h^{(1)} \to ...h^{(l-1)} \to h^{(l)} \to ... \to h^{(L)} \to \hat{Y}, \tag{11.16}$$

where $h^{(l)}$ is a $d^{(l)}$ dimensional vector which is defined recursively by

$$h^{(l)} = f^{(l)}(W^{(l)}h^{(l-1)} + b^{(l)}), \tag{11.17}$$

for $l = 1, ..., L$. We may treat X as $h^{(0)}$, and $\hat{Y}$ as $h^{(L+1)}$ and $\theta = (W^{(l)}, b^{(l)}, l = 1, ..., L+1)$. $W^{(l)}$ is the weight matrix and $b^{(l)}$ is the bias or intercept vector at layer l. $f^{(l)}$ is element-wise transformation, i.e., for $v = (v_1, ..., v_d)^\top$, $f^{(l)}(v) = (f^{(l)}(v_1), ..., f^{(l)}(v_d))^\top$.

Compared to the discriminative models in the previous section, we now have multiple layers of features $(h^{(l)}, l = 1, ..., L)$. They are recursively defined via (11.17), and they are to be learned from the training data instead of being designed.

For classification, suppose there are K categories, the conditional probability of category k given input X is given by the following soft-max probability:

$$\Pr(Y = k \mid X) = \frac{f_{\theta_k}(X)}{\sum_{k=1}^{K} f_{\theta_k}(X)}, \tag{11.18}$$

where $f_{\theta_k}(X)$ is the score for category k. We may take $f_{\theta_k}(X) = h^{(L)\top}W_k^{(L+1)} + b_k^{(L+1)}$. This final classification layer is usually called the soft-max layer.

The most commonly used non-linear rectification in modern neural nets is the Rectified Linear Unit (ReLU) [109]: $f^{(l)}(a) = \max(0, a)$. The resulting function $f_\theta(X)$ can be considered a multi-dimensional linear spline, i.e., a piecewise linear function. Recall a one-dimensional linear spline is of the form $f(x) = b + \sum_{k=1}^{d} w_k \max(0, x - a_k)$, where a_k are the knots. At each knot a_k, the linear spline takes a turn and changes its slope by w_k. With enough knots, $f(x)$ can approximate any non-linear continuous function. We can view this $f(x)$ as a simplified two-layer network, with $h_k = \max(0, x - a_k)$. The basis function $\max(0, x - a_k)$ is two-piece linear function with a bending at a_k. For multi-dimensional input X, a two-layer network with one-dimensional output is of the following form $f(X) = b^{(2)} + \sum_{k=1}^{d} W_k^{(2)} h_k^{(1)}$, where $h_k^{(1)} = \max(0, W_k^{(1)}X + b_k^{(1)})$, and $W_k^{(1)}$ is the k-th row of $W^{(1)}$. The basis function $\max(0, W_k^{(1)}X + b_k^{(1)})$ is again a two-piece linear function with a bending along the line $W_k^{(1)}X + b_k^{(1)} = 0$. The dividing lines $\{W_k^{(1)}X + b_k^{(1)} = 0, k = 1, ..., d^{(1)}\}$ partition the domain of X into up to $2^{d^{(1)}}$ pieces, and $f(X)$ is a continuous piecewise linear function over these pieces.

In the multi-layer network, the hierarchical layers of $\{h^{(l)}, l = 1, ..., L\}$ partition the domain of X recursively, creating a piecewise linear function with exponentially many pieces [147]. Such reasoning also applies to other forms of rectification functions $f^{(l)}$, as long as they are non-linear and create bending. This makes the neural network an extremely powerful machine for function approximation and interpolation. The recursive partition in neural nets is similar to CART and MARS, but is more flexible.

Back-propagation. Both $\partial f_\theta(X)/\partial\theta$ and $\partial f_\theta(X)/\partial X$ can be computed by the chain-rule back-propagation, and they share the computation of $\partial h^{(l)}/\partial h^{(l-1)} = f^{(l)\prime}(W^{(l)}h^{(l-1)} + b^{(l)})W^{(l)}$ in the chain rule. Because $f^{(l)}$ is element-wise, $f^{(l)\prime}$ is a diagonal matrix.

A recent invention [76] is to reparametrize the mapping (11.17) by $h^{(l)} = h^{(l-1)} + f^{(l)}(W^{(l)}h^{(l-1)} + b^{(l)})$, where $f^{(l)}(W^{(l)}h^{(l-1)} + b^{(l)})$ is used to model the residual term. This enables the learning of very deep networks. One may think of it as modeling an iterative algorithm where the layers l can be interpreted as time steps of the iterative algorithm.

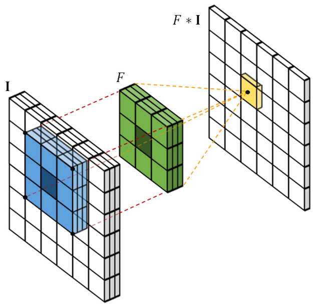


Figure 11.8: Filtering or convolution: applying a filter of the size $3 \times 3 \times 3$ on an image of the size $6 \times 6 \times 3$ to get a filtered image or feature map of 6×6 (with proper boundary handling). Each pixel of the filtered image is computed by the weighted sum of the $3 \times 3 \times 3$ pixels of the input image centered at this pixel. There are 3 color channels (R, G, B), so both the input image and the filter are three-dimensional.

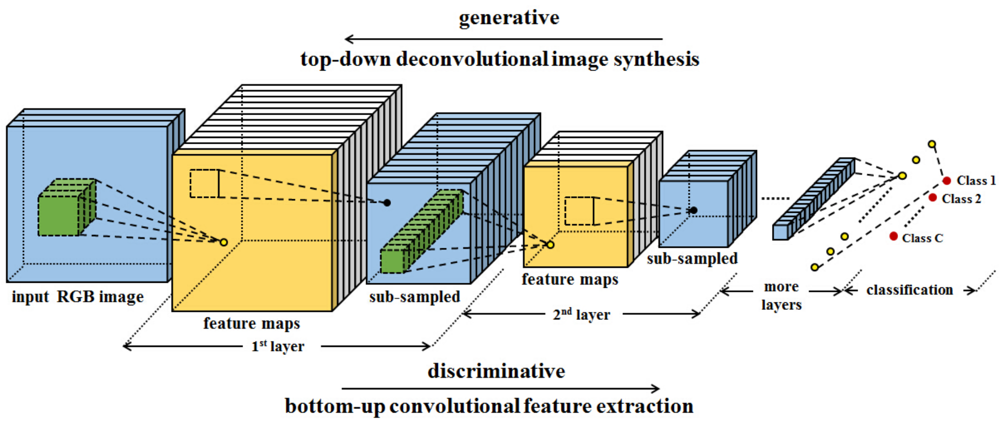


Figure 11.9: Convolutional neural networks consist of multiple layers of filtering and sub-sampling operations for bottom-up feature extraction, resulting in multiple layers of feature maps and their sub-sampled versions. The top layer features are used for classification via multinomial logistic regression. The discriminative direction is from image to category, whereas the generative direction is from category to image.

Convolution. The signal X can be an image, and the linear transformations at each layer may be convolutions with localized kernel functions (i.e. filters). That is, the row vectors of $W^{(l)}$ (as well as the elements of $b^{(l)}$) form different groups, and the vectors in the same group are localized and translation invariant versions of each other, like wavelets. Each group of vectors corresponds to a filter or a kernel or a channel. See

Figures 11.8 and 11.9 for illustrations. Recent networks mostly use small filters of the size 3×3 [173, 179]. The minimal size 1×1 is also a popular choice [117, 179]. Such a filter fuses the features of different channels at the same location, and is often used for reducing or increasing the number of channels. When computing the filtered image, we can also sub-sample it by, e.g., taking one filter response every two pixels. The filter is said to have stride 2.

11.4.3 Descriptive models

This subsection describes the hierarchical form of the descriptive model and the maximum likelihood learning algorithm.

We can generalize the descriptive model in the previous sections to a hierarchical form with multiple layers of features [28, 140, 208, 210],

$$X \to h^{(1)} \to \ldots \to h^{(L)} \to f_\theta(X) \tag{11.19}$$

which is a bottom-up process for computing $f_\theta(X)$, and θ collects all the weight and bias parameters at all the layers. The probability distribution is

$$p_\theta(X) = \frac{1}{Z(\theta)} \exp\left[f_\theta(X)\right] p_0(X), \tag{11.20}$$

where again $p_0(X)$ is the reference distribution such as Gaussian white noise model $p_0(X) \propto \exp\left(-\|X\|^2/2\sigma^2\right)$. Again the normalizing constant is $Z(\theta) = \int \exp(f_\theta(X))p_0(X)dX = \mathrm{E}_{p_0}[\exp(f_\theta(X))]$. The energy function is

$$U_\theta(X) = \|X\|^2/2\sigma^2 - f_\theta(X). \tag{11.21}$$

$q_0(X)$ can also be a uniform distribution within a bounded range, then $U_\theta(X) = -f_\theta(X)$.

The model (11.20) can be considered a hierarchical generalization of the FRAME model. While the energy function of the FRAME model is defined in terms of element-wise non-linear functions of filter responses, model (11.20) involves recursions of this structure at multiple layers according to the ConvNet.

Suppose we observe training examples $\{X_i, i = 1, ..., n\}$. The maximum likelihood learning seeks to maximize $\mathcal{L}(\theta) = \frac{1}{n}\sum_{i=1}^n \log p_\theta(X_i)$. The gradient of the $\mathcal{L}(\theta)$ is

$$\mathcal{L}'(\theta) \quad = \quad \frac{1}{n}\sum_{i=1}^n \frac{\partial}{\partial\theta} f_\theta(X_i) - \mathrm{E}_\theta\left[\frac{\partial}{\partial\theta} f_\theta(X)\right], \tag{11.22}$$

where E_θ denotes the expectation with respect to $p_\theta(X)$. The key identity underlying equation (11.22) is $d\log Z(\theta)/d\theta = \mathrm{E}_\theta[\partial f_\theta(X)/\partial\theta]$.

The expectation in equation (11.22) is analytically intractable and has to be approximated by MCMC, such as the Langevin dynamics, which samples from $p_\theta(X)$ by iterating the following step:

$$X_{\tau+1} = X_\tau - \frac{s^2}{2}\left[\frac{X_\tau}{\sigma^2} - \frac{\partial}{\partial X} f_\theta(X_\tau)\right] + s\mathcal{E}_\tau, \tag{11.23}$$

where τ indexes the time steps of the Langevin dynamics, s is the step size, and $\mathcal{E}_\tau \sim \mathrm{N}(0, I_p)$ is the Gaussian white noise term. A Metropolis-Hastings step can be added to correct for the finiteness of s. The Langevin dynamics was used by [221] for sampling from the linear form of the descriptive model such as the FRAME model.

We can run $\tilde{n}$ parallel chains of Langevin dynamics according to (11.23) to obtain the synthesized examples $\{\tilde{X}_i, i = 1, ..., \tilde{n}\}$. The Monte Carlo approximation to $\mathcal{L}'(\theta)$ is

$$\mathcal{L}'(\theta) \approx \frac{\partial}{\partial\theta}\left[\frac{1}{n}\sum_{i=1}^{n} f_\theta(X_i) - \frac{1}{\tilde{n}}\sum_{i=1}^{\tilde{n}} f_\theta(\tilde{X}_i)\right], \tag{11.24}$$

which is the difference between the observed examples and the synthesized examples. We can then update $\theta^{(t+1)} = \theta^{(t)} + \eta_t \mathcal{L}'(\theta^{(t)})$, with $\mathcal{L}'(\theta^{(t)})$ computed according to (11.24). η_t is the learning rate. The convergence of this algorithm has been studied by [155, 214].

The learning and sampling algorithm is again an "analysis by synthesis" scheme. The sampling step runs the Langevin dynamics by computing $\partial f_\theta(X)/\partial X$, and the learning step updates θ by computing $\partial f_\theta(X)/\partial\theta$. Both derivatives can be computed by back-propagation, and they share the same computations of $\partial h^{(l)}/\partial h^{(l-1)}$.

11.4.4 Introspective learning

This subsection describes the introspective learning method that learns the descriptive model by turning it into a discriminative model.

Model (11.20) corresponds to a classifier in the following sense [28, 94, 208]. Suppose there are K categories, $p_{\theta_k}(X)$, for $k = 1, ..., K$, in addition to the background category $p_0(X)$. The ConvNets $f_{\theta_k}(X)$ for $k = 1, ..., K$ may share common lower layers except the final layer for computing $f_{\theta_k}(X)$. Let ρ_k be the prior probability of category k, $k = 0, ..., K$. Then the posterior probability for classifying an example X to the category k is a soft-max multi-class classifier, i.e., the multinomial logistic regression:

$$\Pr(k|X) = \frac{\exp(f_{\theta_k}(X) + b_k)}{\sum_{k=0}^{K} \exp(f_{\theta_k}(X) + b_k)}, \tag{11.25}$$

where $b_k = \log(\rho_k/\rho_0) - \log Z(\theta_k)$, and for $k = 0$, $f_{\theta_0}(X) = 0$, $b_0 = 0$. Conversely, if we have the soft-max classifier (11.25), then the distribution of each category is $p_{\theta_k}(X)$ of the form (11.20). Thus the descriptive model directly corresponds to the commonly used discriminative ConvNet model.

In the case where we only observe unlabeled examples, we may model them by a single distribution $p_1(X) = p_\theta(X)$ in (11.20), and treat it as the positive distribution, and treat $p_0(X)$ as the negative distribution. Let ρ be the prior probability that a random example comes from p_1. Then the posterior probability that a random example X comes from p_1 is

$$\Pr(1|X) = \frac{1}{1 + \exp[-(f_\theta(X) + b)]}, \tag{11.26}$$

where $b = \log(\rho/(1-\rho)) - \log Z(\theta)$, i.e., a logistic regression.

Generalizing [185], [94] developed an introspective learning method for updating the model by learning a classifier or logistic regression to distinguish between the observed $\{X_i\}$ and the synthesized $\{\tilde{X}_i\}$, and tilt the current model according to the logistic regression. It is also an "analysis by synthesis" scheme as well as an adversarial scheme, except that the analysis is performed by a classifier. Specifically, let $p_t(X)$ be the current model. Each iteration of the introspective learning is as follows. The sampling step generates synthesized examples $\{\tilde{X}_i, i = 1, ..., n\}$ from $p_t(X)$. The learning step fits a logistic regression to separate the real examples $\{X_i, i = 1, ..., n\}$ from the synthesized examples $\{\tilde{X}_i, i = 1, ..., n\}$ to estimate $f_\theta(X)$ and b. Then we let $p_{t+1}(X) = \exp(f_\theta(X))p_t(X)/Z$, where $\log Z = -b$. [94, 185] show that p_t converges to P_{data} if the ConvNet is of infinite capacity. See Figure 11.10 for an illustration.

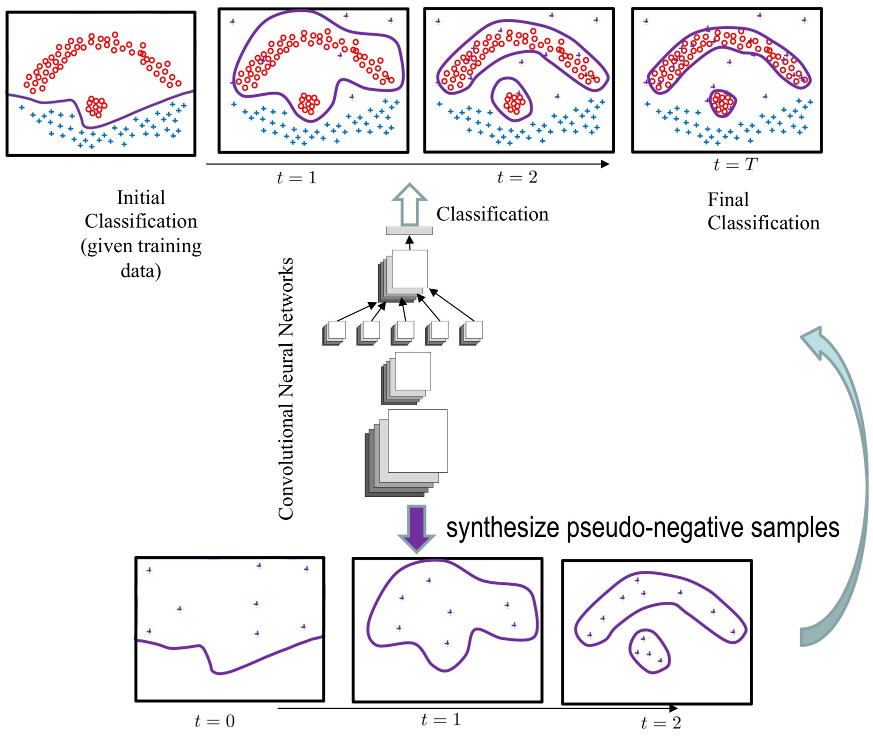

Initial Classification (given training data)
$t = 1$
$t = 2$
$t = T$
Final Classification
Classification
Convolutional Neural Networks
synthesize pseudo-negative samples
$t = 0$
$t = 1$
$t = 2$

Figure 11.10: Introspective learning: The discriminative ConvNet trained on the observed examples versus the synthesized examples generated by the current model can be used to update the model and to generate new examples from the updated model.

Numerical experiments in [94] show that the introspective method learns more accurate classifiers than purely discriminative methods in supervised learning, especially when the size of the training dataset is small or moderate. Figure 11.12 shows the results.

The introspective learning unifies the discriminative model and the descriptive model. Unlike the generative adversarial networks (GAN) [64], the learned classifier is capable of introspection itself: it can be translated into a descriptive model to generate synthesized examples, without relying on a separate generative model.

11.4.5 Generative models

This subsection describes the hierarchical form of the generative model and the maximum likelihood learning algorithm.

We can generalize the generative model in the previous sections to a hierarchical form with multiple layers of hidden variables

$$h \to h^{(L)} \to ... \to h^{(1)} \to X, \tag{11.27}$$

which is a top-down process that transforms h to X. In the linear form of the generative model in the previous sections, the mapping from h to X is linear. In the hierarchical version, the mapping from h to X is a ConvNet defined by

$$h^{(l-1)} = g^{(l)}(W^{(l)}h^{(l)} + b^{(l)}), \tag{11.28}$$

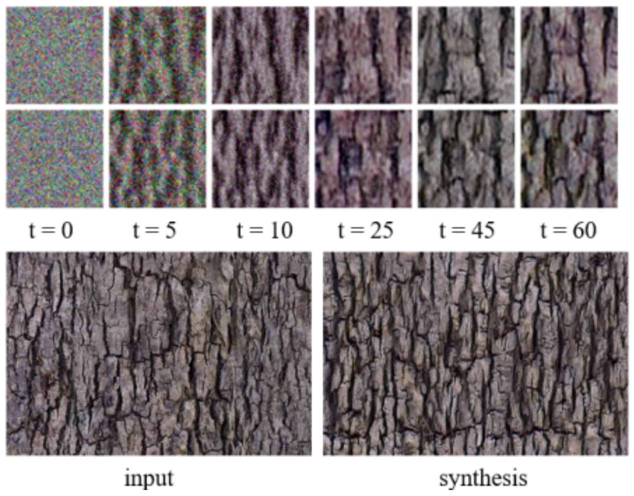


Figure 11.11: Top row: patches of synthesized images in the introspective learning process. Bottom row: Left is the observed image. Right is the synthesized image generated by the learned model.

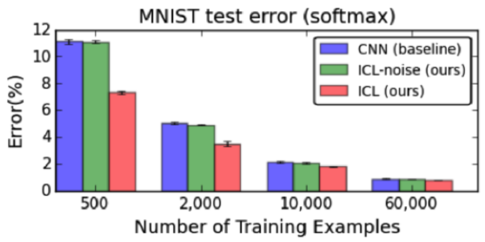


Figure 11.12: Introspective learning improves the classification performances, especially if the training dataset is of small or moderate size.

for $l = L+1, ..., 1$, where $h^{(L+1)} = h$ and $h^{(0)} = X$. $g^{(l)}$ is the non-linear rectification function such as ReLU that is applied element-wise. Let the resulting ConvNet be $X = g_\alpha(h)$, where $\alpha = (W^{(l)}, b^{(l)}, l = 1, 2, ..., L+1)$.

The top-down generative ConvNet was used by [216] to visualize the bottom-up ConvNet learned by the discriminative model. It was also used by [40] to learn a generative model of images of chairs, where the learning is supervised in that for each observed image of chair, a latent vector h is provided to specify the type of chair (represented by a one-hot vector) as well as view point and other geometric properties. The top-down ConvNet can learn accurate mapping from h to X, and the linear interpolation in the space of h leads to very realistic non-linear interpolation in the space of X.

The generative model can also be learned in unsupervised setting where h is unknown [64, 102, 133, 150, 154]. The model has the following form :

$$h \sim \mathrm{N}(0, I_d); \tag{11.29}$$

$$X = g_\alpha(h) + \epsilon; \epsilon \sim \mathrm{N}(0, \sigma^2 I_p), \tag{11.30}$$

where h is the d-dimensional hidden vector of latent factors, $g_\alpha(h)$ is a top-down ConvNet that maps the d-dimensional vector h to the p-dimensional signal X, where $d \leq p$. The model (11.30) is a generalization of factor analysis. While independent component analysis, sparse coding etc. generalize the prior distribution of factor analysis, the model (11.29) and (11.30) maintains the simple prior distribution of factor analysis, but generalizes the linear mapping in factor analysis to non-linear mapping parametrized by the top-down ConvNet (11.28). Like the word to vector representation [131], the hidden vector h may capture semantically meaningful information in the signal X.

The joint distribution

$$\log q_\alpha(h, X) = \log [q(h) q_\alpha(X|h)] \tag{11.31}$$

$$= -\frac{1}{2\sigma^2}\|X - g_\alpha(h)\|^2 - \frac{1}{2}\|h\|^2 + \text{const.} \tag{11.32}$$

The marginal distribution $q_\alpha(X) = \int q_\alpha(h, X)dh$. The posterior distribution of the latent factors $q_\alpha(h|X) = q_\alpha(h, X)/q_\alpha(X) \propto q_\alpha(h, X)$. Here we use the notation q_α to denote the generative model in order to differentiate it from the descriptive model p_θ.

In our recent work [72], we study a maximum likelihood algorithm for learning the generative model (11.30) without resorting to an assisting network. Specifically, if we observe a training set of examples $\{X_i, i = 1, ..., n\}$, then each X_i has a corresponding latent h_i. We can train the generative model by maximizing the observed-data log-likelihood $\mathcal{L}(\alpha) = \frac{1}{n}\sum_{i=1}^n \log q_\alpha(X_i)$.

The gradient of $\mathcal{L}(\alpha)$ can be calculated according to the following identity:

$$\frac{\partial}{\partial \alpha}\log q_\alpha(X) = \frac{1}{q_\alpha(X)}\int \left[\frac{\partial}{\partial \alpha}\log q_\alpha(h, X)\right] q_\alpha(h, X)dh$$

$$= \mathrm{E}_{q_\alpha(h|X)}\left[\frac{\partial}{\partial \alpha}\log q_\alpha(X|h)\right]. \tag{11.33}$$

The expectation with respect to $q_\alpha(h|X)$ can be approximated by drawing samples from $q_\alpha(h|X)$ and then computing the Monte Carlo average.

The Langevin dynamics for sampling h from $p_\alpha(h|X)$ is

$$h_{\tau+1} = h_\tau + \frac{s^2}{2}\left[\frac{1}{\sigma^2}(X - g_\alpha(h_\tau))\frac{\partial}{\partial h}g_\alpha(h_\tau) - h_\tau\right] + s\mathcal{E}_\tau, \tag{11.34}$$

where τ denotes the time step, s is the step size, and $\mathcal{E}_\tau \sim \mathrm{N}(0, I_d)$. Again we can add Metropolis-Hastings step to correct for the finiteness of s.

We can use stochastic gradient algorithm of [214] for learning, where in each iteration, for each X_i, h_i is sampled from $q_\alpha(h_i|X_i)$ by running a finite number of steps of Langevin dynamics starting from the current value of h_i. With the sampled $\{h_i\}$, we can update the parameters α based on the gradient $\mathcal{L}'(\alpha)$, whose Monte Carlo approximation is:

$$\mathcal{L}'(\alpha) \approx \frac{1}{n}\sum_{i=1}^n \frac{\partial}{\partial \alpha}\log q_\alpha(X_i|h_i) = \frac{1}{n}\sum_{i=1}^n \frac{1}{\sigma^2}(X_i - g_\alpha(h_i))\frac{\partial}{\partial \alpha}g_\alpha(h_i). \tag{11.35}$$

It is a non-linear regression of X_i on h_i. We update $\alpha^{(t+1)} = \alpha^{(t)} + \eta_t \mathcal{L}'(\alpha^{(t)})$, with $\mathcal{L}'(\alpha^{(t)})$ computed according to (11.35). η_t is the learning rate. The convergence of this algorithm follows [214].

Alternating back-propagation: The maximum likelihood learning of the generative model (11.30) also follows the alternative back-propagation scheme. The Langevin dynamics for inference needs to compute $\partial g_\alpha(h)/\partial h$. The learning step needs to compute $\partial g_\alpha(h)/\partial \alpha$. Both gradients can be computed by back-propagation and they share the computations of $\partial h^{(l-1)}/\partial h^{(l)}$.

11.5 Variational, adversarial and cooperative learning

Both the descriptive model and the generative model involve intractable integrals. In the descriptive model, the normalizing constant is intractable. In the generative model, the marginal distribution of the observed signal is intractable. Consequently, the maximum likelihood learning algorithms of both models require MCMC sampling such as Langevin dynamics. To learn the descriptive model, we need to sample the synthesized examples. To learn the generative model, we need to sample the latent variables. It is possible to avoid MCMC sampling by variational and adversarial learning. It is also possible to speed up MCMC sampling by cooperative learning.

11.5.1 Variational auto-encoder

This subsection describes the variational learning of the generative model, where an inference model is learned to replace the MCMC sampling of the latent variables.

The maximum likelihood learning of the generative model seeks to minimize $\text{KL}(P_{\text{data}}(X)||q_\alpha(X))$, where $q_\alpha(X) = \int q(h)q_\alpha(X|h)dh$ is the marginal distribution that is intractable. The variational auto-encoder (VAE) [102, 133, 154] changes the objective to

$$\min_\alpha \min_\phi \text{KL}(P_{\text{data}}(X)\rho_\phi(h|X)||q(h)q_\alpha(X|h)), \tag{11.36}$$

where $\rho_\phi(h|X)$ is an analytically tractable approximation to $q_\alpha(h|X)$, and is called the inference model with parameter ϕ. Compared to the maximum likelihood objective $\text{KL}(P_{\text{data}}(X)||q_\alpha(X))$, which is the KL-divergence between the marginal distributions of X, the VAE objective is the KL-divergence between the joint distributions of (h, X), i.e., $P_{\text{data}}(X)\rho_\phi(h|X)$ and $q_\alpha(h, X) = q(h)q_\alpha(X|h)$, which is tractable because it does not involve the marginal $q_\alpha(X)$. The VAE objective is an upper bound of the maximum likelihood objective

$$\text{KL}(P_{\text{data}}(X)\rho_\phi(h|X)||q_\alpha(h, X)) = \text{KL}(P_{\text{data}}(X)||q_\alpha(X)) + \text{KL}(\rho_\phi(h|X)||q_\alpha(h|X)). \tag{11.37}$$

The accuracy of the VAE objective as an approximation to the maximum likelihood objective depends on the accuracy of the inference model $\rho_\phi(h|X)$ as an approximation to the true posterior distribution $q_\alpha(h|X)$.

For simplicity, write $P_{\text{data}}(h, X) = P_{\text{data}}(X)\rho_\phi(h|X)$, where P_{data} here is understood as the distribution of the complete data (h, X), with h imputed by $\rho_\phi(h|X)$, and $Q(h, X) = q(h)q_\alpha(X|h)$. The VAE is

$$\min_\alpha \min_\phi \text{KL}(P_{\text{data}}|Q). \tag{11.38}$$

We can think of VAE from the perspective of alternating projection. (1) Fix α, find ϕ by minimizing $\text{KL}(P_{\text{data}}||Q)$. This is to project the current Q onto the family of P_{data}. (2) Fix ϕ, find α by minimizing $\text{KL}(P_{\text{data}}||Q)$. This is to project the current P_{data} onto the family of Q. Compared to the EM algorithm, projection (1) corresponds to the E-step to impute the missing data in the form of $\rho_\phi(h|X)$, and projection (2) corresponds to the M-step to fit the complete model $q(h)q_\alpha(X|h)$. The basic idea is illustrated by Figure 11.13.

The problem (11.36) is equivalent to maximizing

$$\begin{aligned} & \text{E}_{P_{\text{data}}}\{\text{E}_\phi[\log q_\alpha(h, X)] + \text{entropy}(\rho_\phi(h|X))\} & (11.39) \\ = \; & \text{E}_{P_{\text{data}}}\{\text{E}_\phi[\log q_\alpha(X|h)] - \text{KL}(\rho_\phi(h|X)||q(h))\} & (11.40) \end{aligned}$$

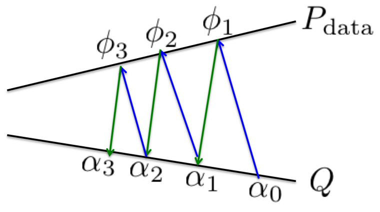


Figure 11.13: VAE as alternating projection, where the straight lines illustrate the families of P_{data} and Q respectively, and each point is a distribution parametrized by ϕ or α.

where E_ϕ denotes the expectation with respect to $\rho_\phi(h|X)$, and $\mathrm{E}_{P_{\text{data}}}$ can be computed by averaging over the training examples. In (11.39) and (11.40), we have $q_\alpha(h, X)$ and $q_\alpha(X|h)$, as a result of merging $q_\alpha(X)$ and $q_\alpha(h|X)$ in (11.37), and both $q_\alpha(h, X)$ and $q_\alpha(X|h)$ are computationally tractable. If $\rho_\phi(h|X) = q_\alpha(h|X)$, then maximizing (11.39) with respect to α becomes the EM algorithm.

One popular choice of $\rho_\phi(h|X)$ is $\mathrm{N}(\mu_\phi(X), \sigma^2_\phi(X))$, where both $\mu_\phi(X)$ and $\sigma^2_\phi(X)$ can be represented by bottom-up neural networks with parameter ϕ.

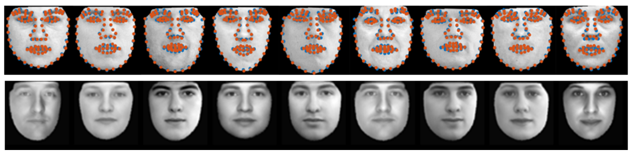


Figure 11.14: Top Row: training images with landmarks. Bottom Row: synthesized images generated by the learned AAM model.

In our recent work [74], we show that VAE can replicate the active appearance model (AAM) [24]. Our experiments were inspired by a recent paper [18] that studied the responses of neurons in specific areas of monkey's brain to face images generated by the AAM model. We show that the observed properties of neurons' responses can be qualitatively replicated by VAE. The AAM model has an explicit shape representation in the form of landmarks, where the landmarks follow a shape model learned by principal component analysis. The faces can be aligned based on the landmarks, and the aligned faces follow an appearance model learned by another principal component analysis. The learning of the shape and appearance models require the landmarks in the training data. Figure 11.14 shows examples of face images to train AAM, and the synthesized face images from the trained AAM.

After learning the AAM model, we generate $20,000$ face images from the learned model. We then learn a VAE model from these images without the landmarks. Figure 11.15 displays test images generated by the AAM model, their corresponding reconstructions by the learned VAE model, and the synthesized images generated by the learned VAE model.

In [18] the authors show that the responses of specific neurons have strong linear relationship with the shape and appearance variables in the original AAM model, where the responses of some neurons are highly correlated with the shape variables while the responses of other neurons are highly correlated with the appearance variables. In fact, one can recover the original face images from the responses of these neurons, by linearly transforming the neurons' responses to the shape and appearance variables of the AAM, and then generating the image by the AAM variables. Apparently the neurons' responses form a code of the input face image that captures both the shape and appearance information of the input image.

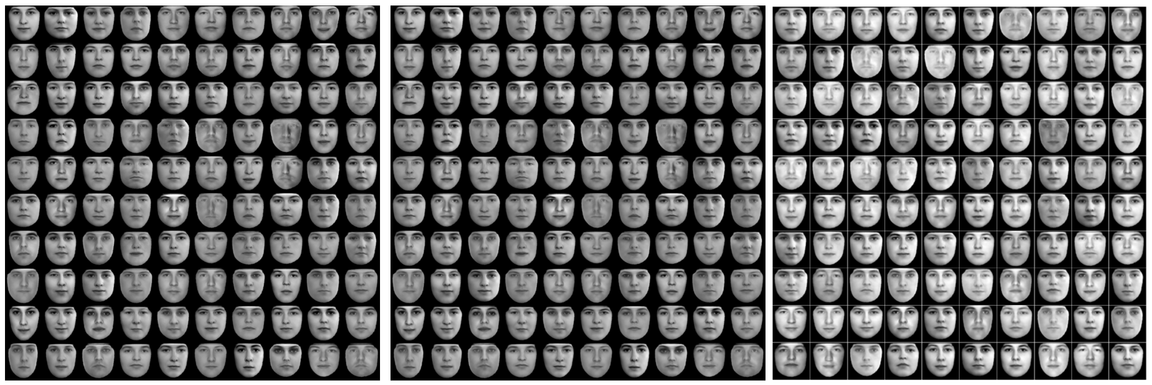


Figure 11.15: Left: test faces generated by AAM. Middle: reconstructed faces by the learned generative model. Right: synthesized images generated by the learned generative model.

We show that the code learned by VAE, i.e., $\mu_\phi(X)$, has very strong linear relationship with the shape and appearance variables in AAM that generates X. The R^2 measure is over 96%. The properties observed by [18] can be qualitatively reproduced by VAE. Even though the AAM model is highly non-linear due to shape deformation, the generative model has no difficulty replicating the AAM model without the supervision in the form of landmarks on the faces.

11.5.2 Adversarial contrastive divergence

This subsection describes the adversarial learning of the descriptive model, where a generative model is learned to replace the MCMC sampling of the descriptive model.

The maximum likelihood learning of the descriptive model seeks to minimize $\text{KL}(P_{\text{data}}(X)||p_\theta(X))$, where the normalizing constant $Z(\theta)$ in p_θ is intractable. Recently [99] and [29] proposed to train the descriptive model p_θ and the generative model q_α jointly, which amounts to modifying the objective to

$$\min_\theta \max_\alpha [\text{KL}(P_{\text{data}}(X)||p_\theta(X)) - \text{KL}(q_\alpha(X)||p_\theta(X))]. \tag{11.41}$$

By maximizing over α, we make $\text{KL}(q_\alpha(X)||p_\theta(X))$ small, so that the objective function in (11.41) is a good approximation to $\text{KL}(P_{\text{data}}||p_\theta)$. Because of the minimax nature of the objective, the learning is adversarial, where θ and α play a minimax game. See Figure 11.16 for an illustration.

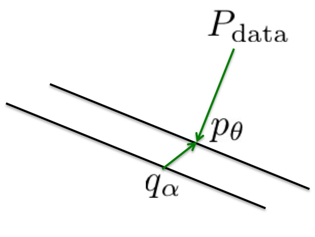


Figure 11.16: Adversarial contrastive divergence. The straight lines illustrate the families of the descriptive and generative models, and each point is a probability distribution. The generative model seeks to approximate the descriptive model, while the descriptive model seeks to get close to the data distribution in contrast to the generative model.

The objective (11.41) contrasts interestingly with the objective for variational learning in (11.37). In the variational objective, we upper bound $\mathrm{KL}(P_{\mathrm{data}}||q_\alpha)$ by adding another KL-divergence, so that we minimize over both α and ϕ. However, in the adversarial objective (11.41), we lower bound $\mathrm{KL}(P_{\mathrm{data}}||p_\theta)$ by subtracting from it another KL-divergence, hence we need to find its saddle point. Thus the sign in front of the second KL-divergence determines whether it is variational learning or adversarial learning.

The adversarial objective (11.41) is also a form of contrastive divergence, except that the synthesized examples are provided by the generative model q_α directly, instead of being obtained by running a finite-step MCMC from the observed examples. We may call (11.41) the adversarial contrastive divergence. It is equivalent to

$$\min_\theta \max_\alpha \left[\mathrm{E}_{P_{\mathrm{data}}}[U_\theta(X)] - \mathrm{E}_\theta[U_\theta(X)] + \mathrm{entropy}(q_\alpha)\right], \tag{11.42}$$

which is the form proposed by [29]. In this form, the $\log Z(\theta)$ term is canceled out, so that we do not have to deal with this intractable term.

However, the entropy term $\mathrm{entropy}(q_\alpha)$ or the second KL-divergence in (11.41) is not in closed form, and still needs approximation. We can again use the variational approach to approximate $\mathrm{KL}(q_\alpha(X)||p_\theta(X))$ by

$$\mathrm{KL}(q_\alpha(X)||p_\theta(X)) + \mathrm{KL}(q_\alpha(h|X)||\rho_\phi(h|X)) = \mathrm{KL}(q_\alpha(h,X)||p_\theta(X)\rho_\phi(h|X)), \tag{11.43}$$

where $\rho_\phi(h|X)$ is again a learned inference model. This leads to the method used by [29]. Again we only need to deal with the tractable joint model $q_\alpha(h,X)$. Thus the learning problem becomes

$$\min_\theta \max_\alpha \max_\phi [\mathrm{KL}(P_{\mathrm{data}}(X)||p_\theta(X)) - \mathrm{KL}(q_\alpha(h,X)||p_\theta(X)\rho_\phi(h|X))]. \tag{11.44}$$

There are three networks that need to be learned, including the descriptive model p_θ, the generative model q_α, and the inference model ρ_ϕ. Write $P_{\mathrm{data}}(h,X) = P_{\mathrm{data}}(X)\rho_\phi(h|X)$, $Q(h,X) = q(h)q_\alpha(X|h)$, and $P(h,X) = p_\theta(X)\rho(h|X)$. The above objective is

$$\min_\theta \max_\alpha \max_\phi [\mathrm{KL}(P_{\mathrm{data}}||P) - \mathrm{KL}(Q||P)]. \tag{11.45}$$

Compared to the variational learning in (11.37), $\rho_\phi(h|X)$ appears on the left side of KL-divergence in (11.37), but it appears on the right side of KL-divergence in (11.44). The learning of $\rho_\phi(h|X)$ is from the synthesized data generated by $q_\alpha(h,X)$ instead of real data. This is similar to the sleep phase of the wake-sleep algorithm [79].

We train the three nets on the down-sampled 32x32 imageNet [33] (roughly 1 million images). For the generative model, starting from the latent vector h of 100 dimensions, we use 5 layers of kernels of stride 2, where the sizes of kernels of the first 4 layers are 4×4, and the size of the kernels of the bottom layer is 3×3. The numbers of channels at these layers are 512, 512, 256, 128, 3 respectively. Each layer is followed by batch normalization and ReLU non-linearity, except the last layer where tanh is used. For the inference model, we use the mirror structure as the generative model. We build the last layer separately to model the posterior mean and variance. For the descriptive model, we use the same structure as the inference net.

Figures 11.17 display the learning results, where the left panel shows randomly selected training examples and the right panel shows the random examples generated by the learned generative model.

11.5.3 Integrating variational and adversarial learning by divergence triangle

We can integrate or unify the variational and adversarial learning methods.

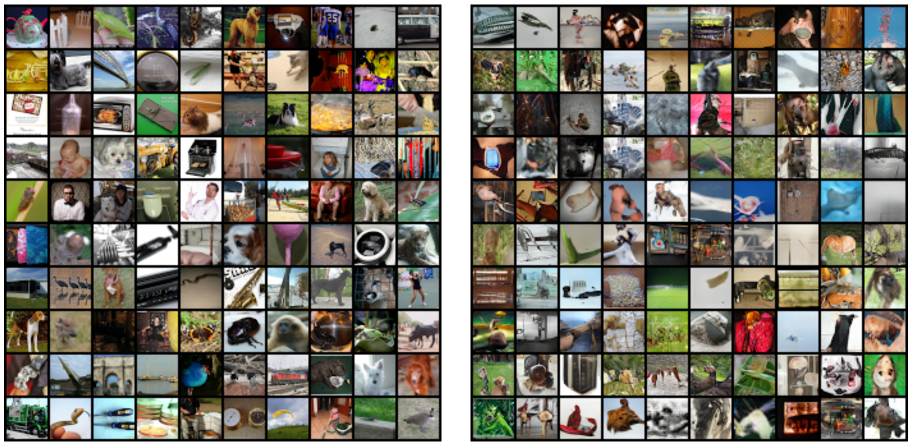


Figure 11.17: Learning the models from the ImageNet dataset. Left: random samples of training examples. Right: synthesized examples generated by the learned generative model.

Following the notation of previous subsections, write $P_{\text{data}}(h, X) = P_{\text{data}}(X)\rho_\phi(h|X)$, $P(h, X) = p_\theta(X)\rho_\phi(h|X)$, and $Q(h, X) = q(h)q_\alpha(X|h)$. It has been noticed by the recent work [73] that the variational objective $\text{KL}(P_{\text{data}}||Q)$ and the adversarial objective $\text{KL}(P_{\text{data}}||P) - \text{KL}(Q||P)$ can be combined into

$$\max_\theta \min_\alpha \min_\phi [\text{KL}(P_{\text{data}}||Q) + \text{KL}(Q||P) - \text{KL}(P_{\text{data}}||P)], \tag{11.46}$$

which is in the form of a divergence triangle formed by P_{data}, P, and Q. One can learn the descriptive model, the generative model, and the inference model jointly using the above objective.

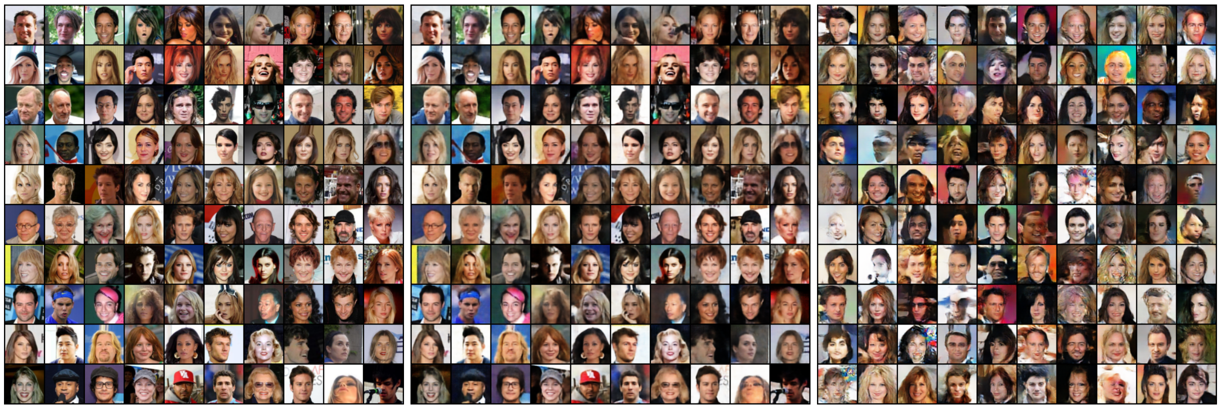


Figure 11.18: Learning the models from CelebA dataset. From left to right: original images, reconstructed images, and generated images.

Figure 11.18 displays an example in [73] where the models are learned from the CelebA dataset. The left panel shows some random training examples, the middle panel shows the corresponding reconstructed examples by the learned inference model, and the right panel shows some examples generated by the learned generative model.

We recruit a layer-wise training scheme to learn models on CelebA-HQ [97] with resolutions of up to $1,024 \times 1,024$ pixels. As in [97], the training starts with down-sampled images with a spatial resolution

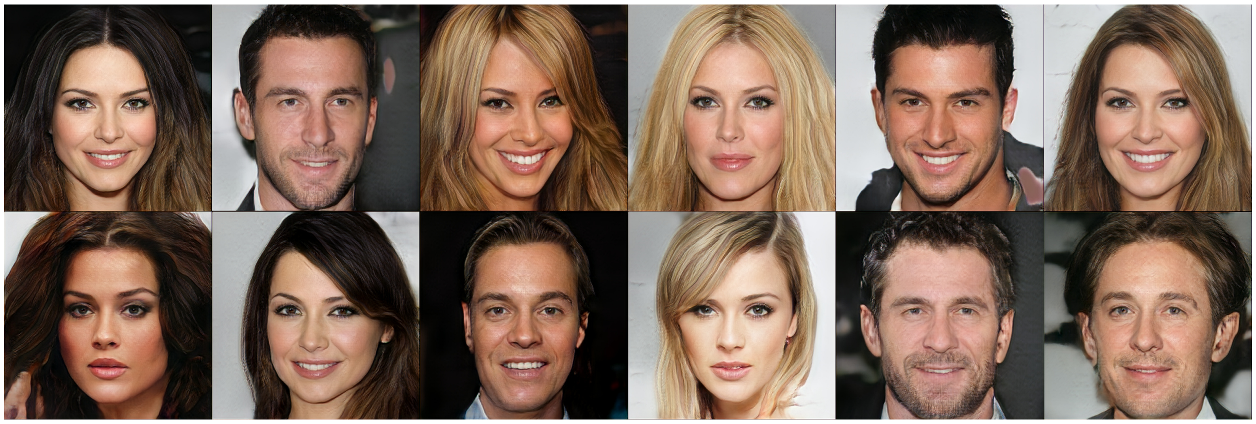


Figure 11.19: Generated samples with $1,024 \times 1,024$ resolution drawn from $g_\theta(z)$ with 512-dimensional latent vector $z \sim N(0, I_d)$ for CelebA-HQ.

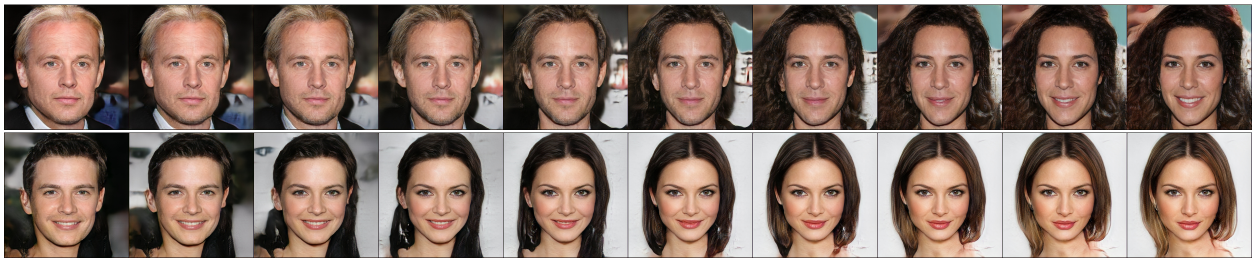


Figure 11.20: High-resolution synthesis from the generator model $g_\theta(z)$ with linear interpolation in latent space (i.e., $(1-\alpha) \cdot z_0 + \alpha \cdot z_1$) for CelebA-HQ.

of 4×4 while progressively increasing the size of the images and number of layers. All three models are grown in synchrony where 1×1 convolutions project between RGB and feature.

Figure 11.19 depicts high-fidelity synthesis in a resolution of $1,024 \times 1,024$ pixels sampled from the generator model $g_\theta(z)$ on CelebA-HQ. Figure 11.20 illustrates linear interpolation in latent space (i.e., $(1-\alpha) \cdot z_0 + \alpha \cdot z_1$), which indicates diversity in the samples.

Therefore, the joint learning in the triangle formulation is not only able to train the three models with stable optimization, but it also achieves synthesis with high fidelity.

11.5.4 Cooperative learning

This subsection describes the cooperative training of the descriptive and generative models which jumpstart each other's MCMC sampling.

We can learn the descriptive model and the generative model separately, and we have been able to scale up the learning to big datasets. However, the separate learning algorithms can still be slow due to MCMC sampling. Recently we discover that we can speed up the learning of the two models by coupling the two maximum likelihood learning algorithms into a cooperative algorithm that we call the CoopNets algorithm [206]. It is based on the following two key observations. (1) The generative model can generate examples directly, so we can use it as an approximate sampler of the descriptive model. (2) The generative model can be learned more easily if the latent factors are known, which is the case with the synthesized examples.

Generative model as a sampler. The generative model can serve as an approximate sampler of the descriptive model. To sample from the descriptive model, we can initialize the synthesized examples by generating examples from the generative model. We first generate $\hat{h}_i \sim \mathrm{N}(0, I_d)$, and then generate $\hat{X}_i = g(\hat{h}_i; \alpha) + \epsilon_i$, for $i = 1, ..., \tilde{n}$. If the current generative model q_α is close to the current descriptive model p_θ, then the generated $\{\hat{X}_i\}$ should be a good initialization for sampling from the descriptive model, i.e., starting from the $\{\hat{X}_i, i = 1, ..., \tilde{n}\}$, we run Langevin dynamics for l steps to get $\{\tilde{X}_i, i = 1, ..., \tilde{n}\}$, which are revised versions of $\{\hat{X}_i\}$. These $\{\tilde{X}_i\}$ can be used as the synthesized examples from the descriptive model. We can then update θ in the same way as we learn the descriptive model.

MCMC teaching. The descriptive model can teach the generative model via MCMC. In order to update α of the generative model, we treat the $\{\tilde{X}_i, i = 1, ..., \tilde{n}\}$ produced by the above procedure as the training data for the generative model. Since these $\{\tilde{X}_i\}$ are obtained by the Langevin dynamics initialized from the $\{\hat{X}_i, i = 1, ..., \tilde{n}\}$, which are generated by the generative model with known latent factors $\{\hat{h}_i, i = 1, ..., \tilde{n}\}$, we can update α by learning from $\{(\hat{h}_i, \tilde{X}_i), i = 1, ..., \tilde{n}\}$, which is a supervised learning problem, or more specifically, a non-linear regression of $\tilde{X}_i$ on $\hat{h}_i$. At $\alpha^{(t)}$, the latent factors $\hat{h}_i$ generates and thus reconstructs the initial example $\hat{X}_i$. After updating α, we want $\hat{h}_i$ to reconstruct the revised example $\tilde{X}_i$. That is, we revise α to absorb the MCMC transition from $\hat{X}_i$ to $\tilde{X}_i$ for sampling the descriptive model, so that the generative model shifts its density from $\{\hat{X}_i\}$ to $\{\tilde{X}_i\}$. The left diagram in (11.47) illustrates the basic idea.

$$\begin{tikzcd} \hat{h}_i \arrow[d, Rightarrow, "\alpha^{(t)}"'] \arrow[dr, Rightarrow, "\alpha^{(t+1)}"] & \\ \hat{X}_i \arrow[r, dashed, "\theta^{(t)}"'] & \tilde{X}_i \end{tikzcd} \qquad \begin{tikzcd} \hat{h}_i \arrow[r, dashed, "\alpha^{(t)}"] \arrow[d, Rightarrow, "\alpha^{(t)}"'] & \tilde{h}_i \arrow[d, Rightarrow, "\alpha^{(t+1)}"] \\ \hat{X}_i \arrow[r, dashed, "\theta^{(t)}"'] & \tilde{X}_i \end{tikzcd} \tag{11.47}$$

In the two diagrams in (11.47), the double-line arrows indicate generation and reconstruction by the generative model, while the dashed-line arrows indicate Langevin dynamics for MCMC sampling and inference in the two models. The diagram on the right in (11.47) illustrates a more rigorous method, where we initialize the Langevin inference of $\{h_i, i = 1, ..., \tilde{n}\}$ from $\{\hat{h}_i\}$, and then update α based on $\{(\tilde{h}_i, \tilde{X}_i), i = 1, ..., \tilde{n}\}$. The diagram on the right shows how the two models jumpstart each other's MCMC.

The learning of the descriptive model is based on the modified contrastive divergence,

$$\text{KL}(P_{\text{data}} \| p_\theta) - \text{KL}(M_\theta q_\alpha \| p_\theta), \tag{11.48}$$

where q_α provides the initialization of the finite-step MCMC, whose transition kernel is denoted M_θ, and $M_\theta q_\alpha$ denotes the marginal distribution obtained after running M_θ from q_α. The learning of the generative model is based on how $M_\theta q_\alpha$ modifies q_α, and is accomplished by $\min_{q_\alpha} \text{KL}(M_\theta q_{\alpha_t} \| q_\alpha)$. In the idealized case of infinite capacity of q_α so that the KL-divergence can be minimized to zero, the learned q_α will satisfy $q_\alpha = M_\theta q_\alpha$, i.e., q_α is the stationary distribution of M_θ. But the stationary distribution of M_θ is nothing but p_θ. Thus the learned q_α will be the same as q_θ. Then the second KL-divergence in (11.48) will become zero, and the learning of the descriptive model is to minimize $\text{KL}(P_{\text{data}} \| p_\theta)$, which is maximum likelihood.

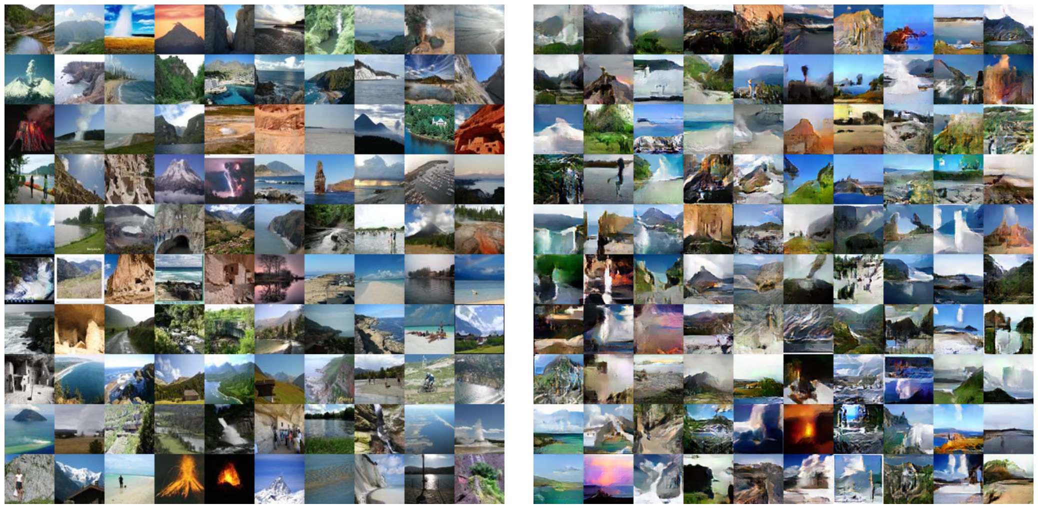


Figure 11.21: Cooperative learning. The training set consists of 11,000 images (64 $\times$ 64) randomly sampled from 10 Imagenet scene categories. Left panel: random samples of training images. Right panel: random samples generated by the learned models.

We conduct experiments on learning from Imagenet dataset [33]. We adopt a 4-layer descriptive model and a 5-layer generative model. We set the number of Langevin dynamics steps in each learning iteration to $l = 10$. The number of learning iterations is $1,000$. After learning the models, we synthesize images using the learned models.

In our first experiment, we learn from images that are randomly sampled from 10 Imagenet scene categories. The number of images sampled from each category is 1100. We mix the images from all the categories as a single training set. Figure 11.21 displays the observed examples randomly sampled from the training set, and the synthesized examples generated by the CoopNets algorithm.

Figure 11.22 shows 4 examples of interpolating between latent vectors h. For each row, the images at the two ends are generated from h vectors randomly sampled from $\text{N}(0, I_d)$. Each image in the middle is obtained by first interpolating the h vectors of the two end images, and then generating the image using the learned models. This experiment shows that we learn smooth generative model that traces the manifold of the data distribution.

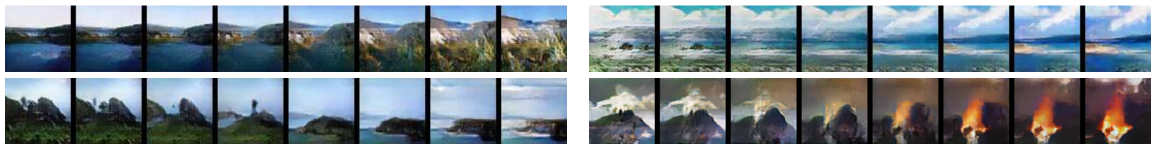


Figure 11.22: Interpolation between latent vectors of the images on the two ends.

We evaluate the synthesis quality by the Inception score [164]. Our method is competitive to DCGAN [150], EBGAN [217], Wasserstein GAN [5], InfoGAN [20], VAE [102], the method of [99].

Compared to the three nets in [29], the cooperative learning method only needs two nets. Moreover, the finite-step MCMC serves to bridge the generative model and the descriptive model, so that the synthesized examples are closer to fair samples from the descriptive model.

11.5.5 Conditional learning via fast thinking initializer and slow thinking solver

Recently, [209] 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

$$\pi_\alpha(x|c) = \frac{1}{Z(c,\alpha)} \exp[f_\alpha(x,c)], \tag{11.49}$$

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 α 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. Both the conditional generator model and the conditional energy-based model can be learned jointly by the cooperative learning scheme.

Figure 11.23 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 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 11.24 shows some examples of pattern completion on the CMP (Center for Machine Perception) Facades data set [189] by learning a mapping from an occluded image (256×256 pixels), where a mask of the size of 128×128 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.

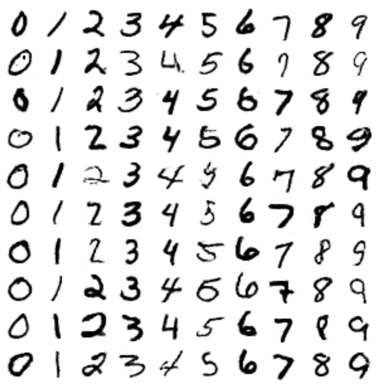


Figure 11.23: Generated handwritten digits conditioned on class labels. Each column is conditioned on one class label, and each row represents a different generated handwritten digit image. The synthesized images are generated by the jointly trained initializer and solver from 30,000 handwritten digit images along with their class labels from the MNIST database. The image size is 64×64 pixels. Abbreviation: MNIST, Modified National Institute of Standards and Technology.

In terms of inverse reinforcement learning, 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 than to specify how to produce what one wants.

11.6 Discussion

To summarize the relationships between the non-hierarchical linear forms and the hierarchical non-linear forms of the three families of models, the non-hierarchical form has one layer of features or hidden variables, and they are designed. The hierarchical form has multiple layers of features or hidden variables, and all the layers are learned from the data.

To summarize the relationships between the three families of models, we have the following connections:

1. The discriminative model and the descriptive model can be translated into each other by the Bayes rule. The introspective learning method unifies the two models.
2. The descriptive model and the generative model can be learned together by adversarial contrastive divergence or the cooperative learning method via MCMC teaching.
3. The discriminative model and the generative model can be learned together by adversarial training.

See Figure 11.25 for an illustration.

Besides the models reviewed in this paper, there are other probabilistic models, such as the deep Boltzmann machine [80, 115, 163], which is an energy-based model with multiple layers of latent variables, auto-regressive models [145], the deep generalizations of the independent component analysis model [36, 37].

In the cooperative learning, the descriptive model and the generative model are parametrized by separate networks. It is more desirable to integrate the two classes of models within a common network.

The existing models are still quite far from what Grenander might have searched for, in that they are still more or less black box models with ConvNet parametrizations. A more interpretable model is the

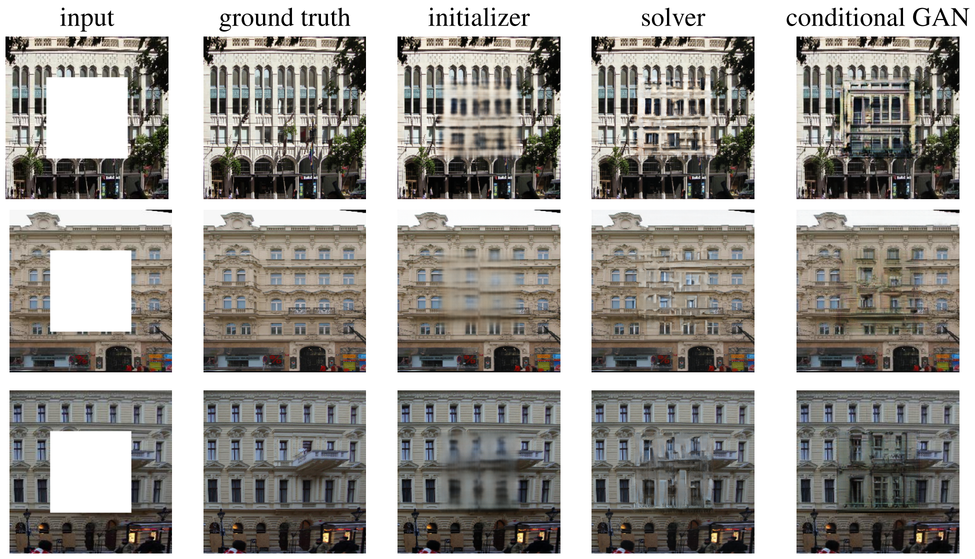


Figure 11.24: Pattern completion by conditional learning. Each row displays one example. The first image is the testing image (256 $\times$ 256 pixels) with a hole of 128 $\times$ 128 that needs to be recovered, the second image shows the ground truth, the third image shows the result recovered by the 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.

And-Or graph [222], which alternates between layers of And nodes and Or nodes. An And node models the composition of parts, while an Or node models the alternative choices of parts according to a certain probability distribution. Such an And-Or grammar can generalize to unseen patterns by reconfiguration of parts. In fact the neural network can be interpreted as a dense version of And-Or graph in that the linear weighted sum can be interpreted as And nodes and the rectification and max pooling can be interpreted as Or nodes. Figure 11.26 shows an example of And-Or template of animal faces [169].

Ideally, as illustrated by Figure 11.27, we should have simple descriptive and generative models at the lowest layers, with the descriptive models accounting for high dimensional or high entropy patterns such as stochastic textures, and the generative models accounting for low-dimensional or low entropy patterns such as textons. In the middle layers we should have stochastic grammars to define the explicit compositional patterns of objects and their parts, as well as their relations [61,222]. At the top layer, we should have logical reasoning based on the learned common sense about physics, funtionality and causality. It is our hope that a unified model of this form can be developed in the future.

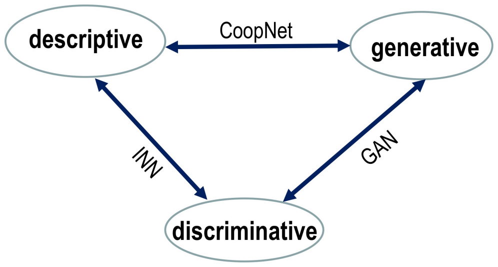


Figure 11.25: The connections between the three families of models. The discriminative and the generative models are connected by the generative adversarial networks (GAN). The discriminative and the descriptive models are connected by the introspective neural networks (INN). The descriptive and the generative models are connected by cooperative learning.

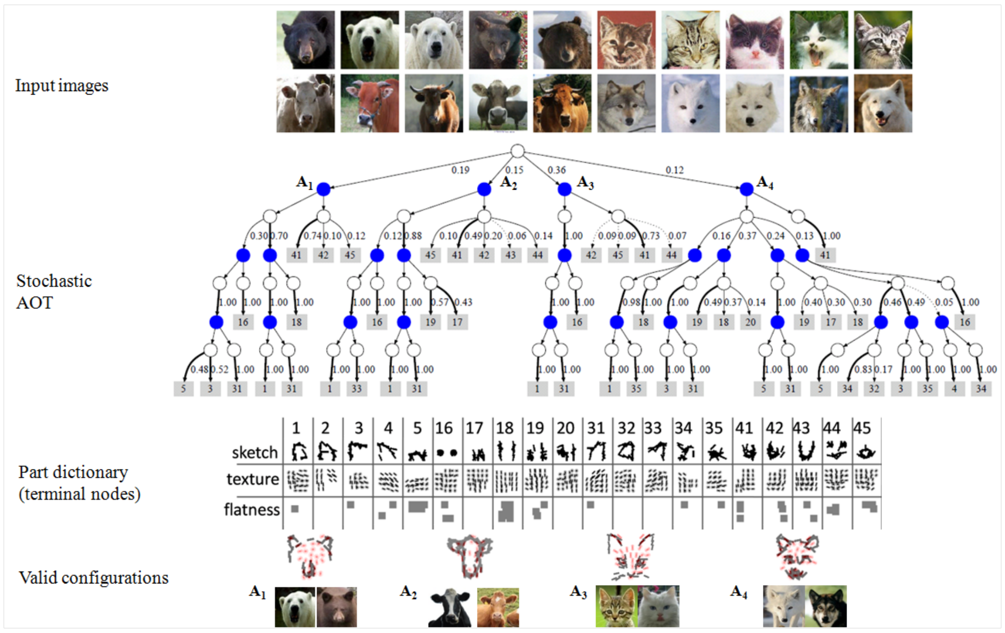


Figure 11.26: And-Or template for modeling recursive compositions of alternative parts: Each And node (blue circle) is a composition of some Or nodes. Each Or node (blank circle) is a probability distribution over some And nodes. An And node models the composition of parts. An Or node models the alternative choices of each part.

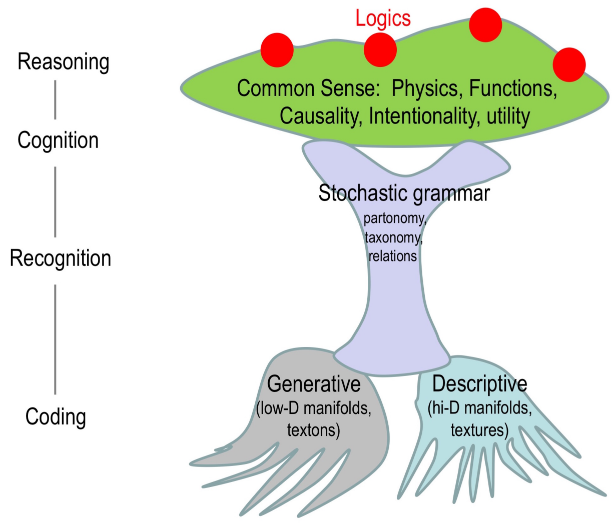

Logics
Reasoning
Cognition
Recognition
Coding
Common Sense: Physics, Functions, Causality, Intentionality, utility
Stochastic grammar
partonomy, taxonomy, relations
Generative
(low-D manifolds, textons)
Descriptive
(hi-D manifolds, textures)

Figure 11.27: Hierarchical representation of patterns, with simple generative and descriptive models for textons and textures at the lower layers, the stochastic grammar in the middle layers, and logic reasoning with common sense at higher layers.

Bibliography

[1] 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.

[2] 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.

[3] Shun-ichi Amari and Hiroshi Nagaoka. *Methods of information geometry*, volume 191. American Mathematical Soc., 2007.

[4] Daniel J. Amit. Modeling brain function: The world of attractor neural networks. *Modeling Brain Function*, 1989.

[5] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan. *arXiv preprint arXiv:1701.07875*, 2017.

[6] 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.

[7] Horace B Barlow. Possible principles underlying the transformations of sensory messages. 1961.

[8] Jens Behrmann, Will Grathwohl, Ricky T. Q. Chen, David Duvenaud, and Jörn-Henrik Jacobsen. Invertible residual networks. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, pages 573–582, 2019.

[9] Peter N Belhumeur. A bayesian approach to binocular steropsis. *International Journal of Computer Vision*, 19(3):237–260, 1996.

[10] Anthony J Bell and Terrence J Sejnowski. The independent components of natural scenes are edge filters. *Vision Research*, 37(23):3327–3338, 1997.

[11] Yoshua Bengio, Ian J. Goodfellow, and Aaron Courville. Deep learning. Book in preparation for MIT Press, 2015.

[12] James R Bergen and EH Adelson. Theories of visual texture perception. *Spatial vision*, 10:114–134, 1991.

[13] Julian Besag. Spatial interaction and the statistical analysis of lattice systems. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 192–236, 1974.

[14] Leo Breiman, Jerome Friedman, Charles J Stone, and Richard A Olshen. *Classification and regression trees*. CRC press, 1984.

[15] 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.

[16] Yue Cao, Bin Liu, Mingsheng Long, and Jianmin Wang. Hashgan: Deep learning to hash with pair conditional wasserstein gan. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[17] 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.

[18] Le Chang and Doris Y Tsao. The code for facial identity in the primate brain. *Cell*, 169(6):1013–1028, 2017.

[19] 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.

[20] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2172–2180, 2016.

[21] LI Chongxuan, Taufik Xu, Jun Zhu, and Bo Zhang. Triple generative adversarial nets. In *Advances in neural information processing systems*, pages 4088–4098, 2017.

[22] Anna Choromanska, Mikael Henaff, Michael Mathieu, Gérard Ben Arous, and Yann LeCun. The loss surface of multilayer networks. *arXiv preprint arXiv:1412.0233*, 2014.

[23] Adam Coates, Andrew Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 215–223, 2011.

[24] 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.

[25] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[26] Thomas M Cover and Joy A Thomas. *Elements of information theory*. John Wiley & Sons, 2012.

[27] George R Cross and Anil K Jain. Markov random field texture models. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, (1):25–39, 1983.

[28] Jifeng Dai, Yang Lu, and Ying Nian Wu. Generative modeling of convolutional neural networks. In *International Conference on Learning Representations*, 2014.

[29] Zihang Dai, Amjad Almahairi, Philip Bachman, Eduard Hovy, and Aaron Courville. Calibrating energy-based generative adversarial networks. In *International Conference on Learning Representations*, 2017.

[30] 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.

[31] 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.

[32] 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.

[33] 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.

[34] 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.

[35] Persi Diaconis and David Freedman. On the statistics of vision: the julesz conjecture. *Journal of Mathematical Psychology*, 24(2):112–138, 1981.

[36] Laurent Dinh, David Krueger, and Yoshua Bengio. NICE: Non-linear independent components estimation. *arXiv preprint arXiv:1410.8516*, 2014.

[37] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real NVP. In *International Conference on Learning Representations*, volume abs/1605.08803, 2017.

[38] Gianluca Donato and Serge Belongie. Approximate thin plate spline mappings. In *European conference on computer vision*, pages 21–31. Springer, 2002.

[39] David Leigh Donoho. Sparse components of images and optimal atomic decompositions. *Constructive Approximation*, 17(3):353–382, 2001.

[40] 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.

[41] 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.

[42] M Elad, M Aharon, and AM Bruckstein. The k-svd: An algorithm for designing of overcomplete dictionaries for sparse representations. *IEEE Transactions On Signal Processing*, 15(12):3736–3745, 2006.

[43] Michael Elad. *Sparse and redundant representations: from theory to applications in signal and image processing*. Springer Science & Business Media, 2010.

[44] 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.

[45] Selim Esedoglu and Riccardo March. Segmentation with depth butwithout detecting junctions. *Journal of Mathematical Imaging and Vision*, 18(1):7–15, 2003.

[46] David J Field. Relations between the statistics of natural images and the response properties of cortical cells. *JOSA A*, 4(12):2379–2394, 1987.

[47] 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.

[48] 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.

[49] Artur Fridman. Mixed markov field, 2000.

[50] Artur Fridman. Mixed markov models, applied mathematics. *Proceedings of the National Academy of Sciences*, 100(14):8092–8096, 2003.

[51] 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.

[52] Jerome H Friedman. Exploratory projection pursuit. *Journal of the American Statistical Association*, 82(397):249–266, 1987.

[53] Jerome H Friedman. Multivariate adaptive regression splines. *The Annals of Statistics*, pages 1–67, 1991.

[54] Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of Statistics*, 29(5):1189–1232, 2001.

[55] King-Sun Fu and Bharat K Bhargava. Tree systems for syntactic pattern recognition. *Computers, IEEE Transactions on*, 100(12):1087–1099, 1973.

[56] 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.

[57] 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.

[58] 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.

[59] 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.

[60] 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.

[61] Stuart Geman, Daniel F Potter, and Zhiyi Chi. Composition systems. *Quarterly of Applied Mathematics*, 60(4):707–736, 2002.

[62] James J Gibson. The perception of the visual world. 1950.

[63] 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.

[64] 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.

[65] Will Grathwohl, Ricky T. Q. Chen, Jesse Bettencourt, Ilya Sutskever, and David Duvenaud. FFJORD: free-form continuous dynamics for scalable reversible generative models. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*, 2019.

[66] Ulf Grenander. A unified approach to pattern analysis. *Advances in Computers*, 10:175–216, 1970.

[67] Ulf Grenander and Michael I Miller. *Pattern theory: from representation to inference*. Oxford University Press, 2007.

[68] Peter Grindrod. *The theory and applications of reaction-diffusion equations: patterns and waves*. Clarendon Press, 1996.

[69] 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.

[70] 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.

[71] 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.

[72] Tian Han, Yang Lu, Song-Chun Zhu, and Ying Nian Wu. Alternating back-propagation for generator network. In *AAAI*, volume 3, page 13, 2017.

[73] Tian Han, Erik Nijkamp, Xiaolin Fang, Song-Chun Zhu, and Ying Nian Wu. Divergence triangle for joint training of energy-based model, generator model and inference model. 2018.

[74] Tian Han, Jiawen Wu, and Ying Nian Wu. Replicating active appearance model by generator network. In *International Joint Conferences on Artificial Intelligence*, 2018.

[75] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning: data mining, inference and prediction*. Springer, 2009.

[76] 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.

[77] 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.

[78] Geoffrey E Hinton. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14(8):1771–1800, 2002.

[79] Geoffrey E Hinton, Peter Dayan, Brendan J Frey, and Radford M Neal. The wake-sleep algorithm for unsupervised neural networks, 1995.

[80] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18(7):1527–1554, 2006.

[81] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[82] 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.

[83] 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.

[84] 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.

[85] Berthold KP Horn. The curve of least energy. *CACM Transactions on Mathematical Software*, 9(4):441–460, 1983.

[86] 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.

[87] Aapo Hyvärinen. Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 6(Apr):695–709, 2005.

[88] Aapo Hyvarinen. Connections between score matching, contrastive divergence, and pseudolikelihood for continuous-valued variables. *IEEE Transactions on neural networks*, 18(5):1529–1531, 2007.

[89] Aapo Hyvärinen, Juha Karhunen, and Erkki Oja. *Independent component analysis*, volume 46. John Wiley & Sons, 2004.

[90] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[91] J.H.Elder and R.M.Goldberg. Contour grouping with prior models. *Journal of Vision*, 2(4):324–353, 2002.

[92] Xu Ji, João F Henriques, and Andrea Vedaldi. Invariant information clustering for unsupervised image classification and segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 9865–9874, 2019.

[93] Zhuxi Jiang, Yin Zheng, Huachun Tan, Bangsheng Tang, and Hanning Zhou. Variational deep embedding: An unsupervised and generative approach to clustering. *arXiv preprint arXiv:1611.05148*, 2016.

[94] Long Jin, Justin Lazarow, and Zhuowen Tu. Introspective learning for discriminative classification. In *Advances in Neural Information Processing Systems*, 2017.

[95] Bela Julesz. Visual pattern discrimination. *IRE Transactions on Information Theory*, 8(2):84–92, 1962.

[96] Bela Julesz et al. Textons, the elements of texture perception, and their interactions. *Nature*, 290(5802):91–97, 1981.

[97] 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.

[98] 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.

[99] Taesup Kim and Yoshua Bengio. Deep directed generative models with energy-based probability estimation. *ICLR Workshop*, 2016.

[100] 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.

[101] 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.

[102] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. *International Conference for Learning Representations*, 2014.

[103] Diederik P. Kingma and Max Welling. Efficient gradient-based inference through transformations between bayes nets and neural nets. *CoRR*, abs/1402.0480, 2014.

[104] Durk P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. In *Advances in Neural Information Processing Systems*, pages 10215–10224, 2018.

[105] Durk P Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. Semi-supervised learning with deep generative models. In *Advances in neural information processing systems*, pages 3581–3589, 2014.

[106] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

[107] 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.

[108] Alex Krizhevsky and Geoffrey E Hinton. Learning multiple layers of features from tiny images. 2009.

[109] 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.

[110] Dmitry Krotov and John J Hopfield. Unsupervised learning by competing hidden units. *Proceedings of the National Academy of Sciences*, page 201820458, 2019.

[111] 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.

[112] 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.

[113] Ann B Lee, JG Huang, and DB Mumford. Random collage model for natural images. *International Journal of Computer Vision*, 2000.

[114] Daniel D Lee and H Sebastian Seung. Algorithms for non-negative matrix factorization. In *Advances in Neural Information Processing Systems*, pages 556–562, 2001.

[115] Honglak Lee, Roger Grosse, Rajesh Ranganath, and Andrew Y Ng. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *International Conference on Machine Learning*, pages 609–616. ACM, 2009.

[116] 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.

[117] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. *arXiv preprint arXiv:1312.4400*, 2013.

[118] 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.

[119] Chuanhai Liu, Donald B Rubin, and Ying Nian Wu. Parameter expansion to accelerate em: The px-em algorithm. *Biometrika*, 85(4):755–770, 1998.

[120] Fayao Liu, Chunhua Shen, and Guosheng Lin. Deep convolutional neural fields for depth estimation from a single image. In *CVPR*, 2015.

[121] Jun S Liu. *Monte Carlo strategies in scientific computing*. Springer Science & Business Media, 2008.

[122] Yang Lu, Song-Chun Zhu, and Ying Nian Wu. Learning FRAME models using CNN filters. In *The AAAI Conference on Artificial Intelligence*, 2016.

[123] Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, Ian Goodfellow, and Brendan Frey. Adversarial autoencoders. *arXiv preprint arXiv:1511.05644*, 2015.

[124] Jitendra Malik. Interpreting line drawings of curved objects, 1986.

[125] 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.

[126] Stéphane G Mallat and Zhifeng Zhang. Matching pursuits with time-frequency dictionaries. *IEEE Transactions on Signal Processing*, 1993.

[127] David Marr. Vision: A computational approach, 1982.

[128] David Marr. Vision: a computational investigation into the human representation and processing of visual information. *WH San Francisco: Freeman and Company*, 1(2), 1982.

[129] 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.

[130] Georges Matheron. *Random sets and integral geometry*. John Wiley & Sons, 1975.

[131] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119, 2013.

[132] Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, Ken Nakae, and Shin Ishii. Distributional smoothing by virtual adversarial examples. *stat*, 1050:2, 2015.

[133] Andriy Mnih and Karol Gregor. Neural variational inference and learning in belief networks. In *ICML*, pages 1791–1799, 2014.

[134] 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.

[135] David Mumford. The statistical description of visual signals. *MATHEMATICAL RESEARCH*, 87:233–256, 1996.

[136] David Mumford and Jayant Shah. Optimal approximations by piecewise smooth functions and variational problems. *Communications on Pure and Applied Mathematics*, 42(5):577–685, 1988.

[137] James D. Murray. A pre-pattern formation mechanism for animal coat markings. *Journal of Theoretical Biology*, 88(1):161–199, 1981.

[138] Radford M Neal et al. Mcmc using hamiltonian dynamics. *Handbook of markov chain monte carlo*, 2(11):2, 2011.

[139] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. 2011(2):5, 2011.

[140] Jiquan Ngiam, Zhenghao Chen, Pang W Koh, and Andrew Y Ng. Learning deep energy models. In *International Conference on Machine Learning*, pages 1105–1112, 2011.

[141] 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.

[142] Mark Nitzberg and David Mumford. The 2.1-d sketch. In *International Conference on Computer Vision (ICCV)*, pages 138–144, 1990.

[143] Mark Nitzberg and Takahiro Shiota. Nonlinear image filtering with edge and corner enhancement. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (8):826–833, 1992.

[144] 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.

[145] Aaron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. *arXiv preprint arXiv:1601.06759*, 2016.

[146] Pentti Paatero and Unto Tapper. Positive matrix factorization: A non-negative factor model with optimal utilization of error estimates of data values. *Environmetrics*, 5(2):111–126, 1994.

[147] 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.

[148] 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.

[149] Bruno Poucet and Etienne Save. Attractors in memory. *Science*, 308(5723):799–800, 2005.

[150] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[151] Antti Rasmus, Mathias Berglund, Mikko Honkala, Harri Valpola, and Tapani Raiko. Semi-supervised learning with ladder networks. In *Advances in Neural Information Processing Systems*, pages 3546–3554, 2015.

[152] 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.

[153] Xiaofeng Ren, Charless C Fowlkes, and Jitendra Malik. Figure/ground assignment in natural images. In *In European Conference on Computer Vision (ECCV)*, pages 614–627, 2006.

[154] 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.

[155] Herbert Robbins and Sutton Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951.

[156] 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.

[157] Kevin Roth, Aurelien Lucchi, Sebastian Nowozin, and Thomas Hofmann. Stabilizing training of generative adversarial networks through regularization. In *Advances in neural information processing systems*, pages 2018–2028, 2017.

[158] 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.

[159] Sam T Roweis and Lawrence K Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, 2000.

[160] Donald B Rubin. *Multiple imputation for nonresponse in surveys*, volume 81. John Wiley & Sons, 2004.

[161] Donald B Rubin and Dorothy T Thayer. EM algorithms for ML factor analysis. *Psychometrika*, 47(1):69–76, 1982.

[162] Daniel L Ruderman. The statistics of natural images. *Network: computation in neural systems*, 5(4):517–548, 1994.

[163] Ruslan Salakhutdinov and Geoffrey E Hinton. Deep boltzmann machines. In *International Conference on Artificial Intelligence and Statistics*, 2009.

[164] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *Advances in Neural Information Processing Systems*, pages 2226–2234, 2016.

[165] Eric Saund. Perceptual organization of occluding contours generated by opaque surfaces. In *Computer Vision and Pattern Recognition (CVPR)*, pages 624–630, 1999.

[166] Eric Saund. Perceptual organization of occluding contours of opaque surfaces. *Computer Vision and Image Understanding*, 76(1):70–82, 1999.

[167] 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.

[168] Claude Elwood Shannon. A mathematical theory of communication. *ACM SIGMOBILE Mobile Computing and Communications Review*, 5(1):3–55, 2001.

[169] Zhangzhang Si and Song-Chun Zhu. Learning and-or templates for object recognition and detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(9):2189–2205, 2013.

[170] Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *European Conference on Computer Vision*, pages 746–760. Springer, 2012.

[171] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354–359, 2017.

[172] 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.

[173] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *ICLR*, 2015.

[174] Steven Skiena. Combinatorics and graph theory with mathematica, 1990.

[175] Casper Kaae Sønderby, Jose Caballero, Lucas Theis, Wenzhe Shi, and Ferenc Huszár. Amortised MAP inference for image super-resolution. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017.

[176] Jost Tobias Springenberg. Unsupervised and semi-supervised learning with categorical generative adversarial networks. *arXiv preprint arXiv:1511.06390*, 2015.

[177] X Yu Stella, 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.

[178] 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.

[179] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2818–2826, 2016.

[180] Yee Whye Teh, Max Welling, Simon Osindero, and Geoffrey E Hinton. Energy-based models for sparse overcomplete representations. *Journal of Machine Learning Research*, 4(Dec):1235–1260, 2003.

[181] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, 58(1):267–288, 1996.

[182] Tijmen Tieleman. Training restricted Boltzmann machines using approximations to the likelihood gradient. In *International Conference on Machine Learning*, pages 1064–1071. ACM, 2008.

[183] Tijmen Tieleman. Training restricted boltzmann machines using approximations to the likelihood gradient. *ICML*, pages 1064–1071, 2008.

[184] Antonio Torralba and Aude Oliva. Depth estimation from image structure. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 24(9):1226–1238, 2002.

[185] Zhuowen Tu. Learning generative models via discriminative approaches. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2007.

[186] 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.

[187] Zhuowen Tu and Song-Chun Zhu. Parsing images into regions, curves and curve groups. *International Journal of Computer Vision*, 69(2):223–249, 2006.

[188] Alan Mathison Turing. The chemical basis of morphogenesis. *Bulletin of mathematical biology*, 52(1-2):153–197, 1990.

[189] 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.

[190] Pascal Vincent. A connection between score matching and denoising autoencoders, 2010.

[191] 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.

[192] David Waltz. Understanding line drawings of scenes with shadows, 1975.

[193] 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.

[194] John YA Wang and Edward H Adelson. Layered representation for motion analysis. In *Computer Vision and Pattern Recognition (CVPR)*, pages 361–366, 1993.

[195] John YA Wang and Edward H Adelson. Representing moving images with layers. *IEEE Trans. Image Processing*, 3(5):625–638, 1994.

[196] Max Welling. Herding dynamical weights to learn. In *International Conference on Machine Learning*, pages 1121–1128. ACM, 2009.

[197] Max Welling, Richard S Zemel, and Geoffrey E Hinton. Self supervised boosting. In *Advances in Neural Information Processing Systems*, pages 665–672, 2002.

[198] Lance R. Willianms and Allen R. Hanson. Perceptual completion of occluded surfaces. *Computer Vision and Image Understanding(CVIU)*, 64(1):1–20, 1996.

[199] 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.

[200] 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.

[201] 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.

[202] 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.

[203] 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.

[204] Ying Nian Wu, Song-Chun Zhu, and Xiuwen Liu. Equivalence of julesz ensembles and frame models. *International Journal of Computer Vision*, 38:247–265, 2000.

[205] 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.

[206] 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.

[207] 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.

[208] Jianwen Xie, Yang Lu, Song-Chun Zhu, and Ying Nian Wu. A theory of generative ConvNet. In *International Conference on Machine Learning*, pages 2635–2644, 2016.

[209] 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.

[210] Jianwen Xie, Song-Chun Zhu, and Ying Nian Wu. Synthesizing dynamic patterns by spatial-temporal generative ConvNet. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 7093–7101, 2017.

[211] Junyuan Xie, Ross Girshick, and Ali Farhadi. Unsupervised deep embedding for clustering analysis. In *International conference on machine learning*, pages 478–487, 2016.

[212] Xianglei Xing, Tian Han, Ruiqi Gao, Song-Chun Zhu, and Ying Nian Wu. Unsupervised disentangling of appearance and geometry by deformable generator network. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10354–10363, June 2019.

[213] 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.

[214] 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.

[215] Fisher Yu, Yinda Zhang, Shuran Song, Ari Seff, and Jianxiong Xiao. LSUN: construction of a large-scale image dataset using deep learning with humans in the loop. *CoRR*, abs/1506.03365, 2015.

[216] Matthew D Zeiler, Graham W Taylor, and Rob Fergus. Adaptive deconvolutional networks for mid and high level feature learning. In *International Conference on Computer Vision*, pages 2018–2025. IEEE, 2011.

[217] Junbo Zhao, Michael Mathieu, and Yann LeCun. Energy-based generative adversarial network. *arXiv preprint arXiv:1609.03126*, 2016.

[218] Qang Zheng and Rama Chellappa. Estimation of illumination direction, albedo, and shape from shading. *IEEE Trans. on PAMI*, 1991.

[219] Bolei Zhou, Agata Lapedriza, Jianxiong Xiao, Antonio Torralba, and Aude Oliva. Learning deep features for scene recognition using places database. In *Advances in Neural Information Processing Systems*, pages 487–495, 2014.

[220] Song-Chun Zhu. Statistical modeling and conceptualization of visual patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(6):691–712, 2003.

[221] Song-Chun Zhu and David Mumford. Grade: Gibbs reaction and diffusion equations. In *ICCV*, pages 847–854. IEEE, 1998.

[222] 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.

[223] 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.

[224] 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.

[225] 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.

[226] Song-Chun Zhu, Ying Nian Wu, and David Mumford. Filters, random fields and maximum entropy (frame): Towards a unified theory for texture modeling. *International Journal of Computer Vision*, 27(2):107–126, 1998.