To appear in IEEE Trans. on PAMI, 2002. A short version appeared in ICCV01.

Image Segmentation by Data-Driven Markov Chain Monte Carlo

Zhuowen Tu and Song-Chun Zhu

Department of Computer and Information Science
Ohio State University
{ztu, szhu}@cis.ohio-state.edu

Abstract

This paper presents a computational paradigm called *Data-Driven Markov Chain Monte Carlo (DDMCMC)* for image segmentation in the Bayesian statistical framework. The paper contributes to image segmentation in four aspects. Firstly, it designs efficient and well balanced Markov Chain dynamics to explore the complex solution space, and thus achieves a nearly global optimal solution independent of initial segmentations. Secondly, it presents a mathematical principle and a *K-adventurers* algorithm for computing multiple distinct solutions from the Markov chain sequence, and thus it incorporates intrinsic ambiguities in image segmentation. Thirdly, it utilizes data-driven (bottom-up) techniques, such as clustering and edge detection, to compute importance proposal probabilities, which drive the Markov chain dynamics and achieve tremendous speedup in comparison to the traditional jump-diffusion methods[12, 11]. Fourthly, the DDMCMC paradigm provides a unifying framework in which the role of many existing segmentation algorithms, such as, edge detection, clustering, region growing, split-merge, snake/balloon, and region competition, are revealed as either realizing Markov chain dynamics or computing importance proposal probabilities. Thus the DDMCMC paradigm combines and generalizes these segmentation methods in a principled way. The DDMCMC paradigm adopts seven parametric and non-parametric image models for intensity and color at various regions. We test the DDMCMC paradigm extensively on both color and grey level images, and some results are reported in this paper.

Keywords: Image segmentation, Markov chain Monte Carlo, Region competition, Data clustering, Edge detection, Markov random field.

1 Introduction

Image segmentation is a long standing problem in computer vision, and it is found difficult and challenging for two main reasons.

The first challenge is the fundamental complexity of modeling a vast amount of visual patterns that appear in generic images. The objective of image segmentation is to parse an image into its constituent components. The latter are various stochastic processes, such as attributed points, lines, curves, textures, lighting variations, and deformable objects. Thus a segmentation algorithm must incorporate many families of image models, and its performance is upper bounded by the accuracy of its image models.

The second challenge is the intrinsic ambiguities in image perception, especially when there is no specific task to guide the attention. Real world images are fundamentally ambiguous, and our perception of an image changes over time. Furthermore, an image often demonstrates details at multiple scales. Thus the more one looks at an image, the more one sees. Therefore, it must be wrong to think that a segmentation algorithm outputs only one result. In our opinion, image segmentation should be considered *a computing process* not a vision *task*. It should output multiple distinct solutions *dynamically* and *endlessly* so that these solutions "best preserve" the intrinsic ambiguity.

Motivated by the above two observations, we present a stochastic computing paradigm called *data-driven Markov chain Monte Carlo* (DDMCMC) for image segmentation. We proceed in five steps.

Firstly, we formulate the problem in a Bayesian/MDL framework[15, 14, 29] with seven families of image models which compete to explain various visual patterns in an image, for example, flat regions, clutter, texture, smooth shading, etc.

Secondly, we decompose the solution space into an union of many subspaces of varying dimensions, and each subspace is a product of a number of subspaces for the image partition and image models (see Figure 3 for a space structure). The Bayesian posterior probability is distributed over such a heterogeneously structured space.

Thirdly, we design ergodic Markov chains to explore the solution space and sample the posterior probability. The Markov chains consist of two types of dynamics: jumps and diffusion. The jump dynamics simulate *reversible* split-and-merge, and model switching. The diffusion dynamics simulate boundary deformation, region growing, region competition[29], and model adaptation. We make the split and merge processes reversible, and the ergodicity and reversibility enable the algorithm to achieve nearly global optimal solution independent of initial segmentation conditions. Thus, this demonstrates major progress over the previous region competition algorithm (Zhu and Yuille, 1996)[29].

Fourthly, we utilize data-driven techniques to guide the Markov chain search, and thus achieves tremendous speedup in comparison to previous MCMC algorithms[10, 12, 11]. In the literature, there are various techniques for improving the Markov chain speed, such as multi-resolution approaches[28, 3], causal Markov models[3, 22], and clustering[28, 27, 2, 9]. In our DDMCMC paradigm, data-driven methods such as edge detection[4] and tracing, data clustering[5, 6] are used. The results of these algorithms are expressed as weighted samples (or particles), which encode non-parametric probabilities in various subspaces. These probabilities respectively approximate the marginal probabilities of the Bayesian posterior probability, and they are used to design importance proposal probabilities to drive the Markov chains.

Fifthly, we propose a mathematical principle and a "K-adventurers" algorithm for selecting and pruning a set of *important* and *distinct* solutions from the Markov chain sequence and at multiple scales of details. The set of solutions encode an approximation to the Bayesian posterior probability. The multiple solutions are computed to minimize a Kullback-Leibler divergence from the approximative posterior to the true posterior, and they preserve the ambiguities in image segmentation.

In summary, the DDMCMC paradigm is about effectively creating particles (by bottom-up clustering/edge detection), composing particles (by importance proposals), and pruning particles (by a K-adventurers algorithm), and these particles represent hypotheses of various grainularities in the solution space.

Conceptually, the DDMCMC paradigm also reveals the roles of some well-known segmentation algorithms. Algorithms such as split-and-merge, region growing, Snake[13] and balloon/bubble[25], region competition[29], and variational methods[14], and PDEs[24] can be viewed as various MCMC jump-diffusion dynamics with minor modifications. Other algorithms, such as edge detection[4] and clustering[6, 8] compute importance proposal probabilities.

We test the algorithm on a wide variety of grey level and color images, and some results are shown in the paper. We also demonstrate multiple solutions and verify the segmentation results by synthesizing (reconstructing) images through sampling the likelihood models.

2 Problem Formulation and Image Models

In this section, we formulate the problem in a Bayesian framework, and discuss the prior and likelihood models that are selected in our experiments.

2.1 The Bayesian Formulation for Segmentation

Let $\Lambda = \{(i,j) : 1 \leq i \leq L, 1 \leq j \leq H\}$ be an image lattice, and $\mathbf{I}_\Lambda$ an image defined on Λ. For any point $v \in \Lambda$, $\mathbf{I}_v \in \{0, ..., G\}$ is the pixel intensity for a grey level image, or $\mathbf{I}_v = (L_v, U_v, V_v)$ for a color image.[1] The problem of image segmentation refers to partitioning the lattice into an unknown number of K disjoint regions

$$\Lambda = \cup_{i=1}^{K} R_i, \quad R_i \cap R_j = \emptyset, \quad \forall i \neq j. \tag{1}$$

Each region $R \subset \Lambda$ needs not to be connected for reason of occlusion. We denote by $\Gamma_i = \partial R_i$ the boundary of R_i. As a slight complication, two notations are used interchangeably in the literature. One treats a region $R \in \Lambda$ as a discrete label map, and the other treats a region boundary $\Gamma(s) = \partial R$ as a continuous contour parameterized by s. The continuous representation is convenient for diffusions while the label map representation is better for maintaining the topology. The level set method[23, 24] provides a good transform between the two.

Each image region $\mathbf{I}_R$ is supposed to be coherent in the sense that $\mathbf{I}_R$ *is a realization from a probabilistic model* $p(\mathbf{I}_R; \Theta)$. Θ represents a stochastic process whose type is indexed by ℓ.

[1]We transfer the (R,G,B) representation to (L^*, u^*, v^*) for better color distance measure.

Thus a segmentation is denoted by a vector of hidden variables W, which describes the world state for generating the image $\mathbf{I}$.

$$W = (K, \{(R_i, \ell_i, \Theta_i);\ \ i = 1, 2, ..., K\}).$$

In a Bayesian framework, we make inference about W from $\mathbf{I}$ over a solution space Ω.

$$W \sim p(W|\mathbf{I}) \propto p(\mathbf{I}|W)p(W),\ \ W \in \Omega.$$

As we mentioned before, the first challenge in segmentation is to obtain realistic image models. In the following, we briefly discuss the prior and likelihood models selected in our experiments.

2.2 The Prior Probability $p(W)$

The prior probability $p(W)$ is a product of the following four probabilities.

1. An exponential model for the number of regions $p(K) \propto e^{-\lambda_0 K}$.

2. A general smoothness Gibbs prior for the region boundaries $p(\Gamma) \propto e^{-\mu \oint_\Gamma ds}$.

3. A model for the size of each region. Recently both empirical and theoretical studies[1, 16] on the statistics of natural images indicate that the size of a region $A = |R|$ in natural images follows a distribution, $p(A) \propto \frac{1}{A^\alpha}, \alpha \sim 2.0$. Such a prior encourages large regions to form. In our experiments, we found this prior is not strong enough to enforce large regions, instead we take a distribution

$$p(A) \propto e^{-\gamma A^c}, \tag{2}$$

where $c = 0.9$ is a constant. γ is a scale factor which controls the scale of the segmentation. This scale factor is in spirit similar to the "clutter factor" found by Mumford and Gidas[20] in studying natural images. It is an indicator for how "busy" an image is. In our experiments, it is typically set to $\gamma = 2.0$ and is the only free parameter in this paper.

4. The prior for the type of model $p(\ell)$ is assumed to be uniform, and the prior for the parameters Θ of an image model penalizes model complexity in terms of the number of parameters Θ, $p(\Theta|\ell) \propto e^{-\nu|\Theta|}$.

In summary, we have the following prior model

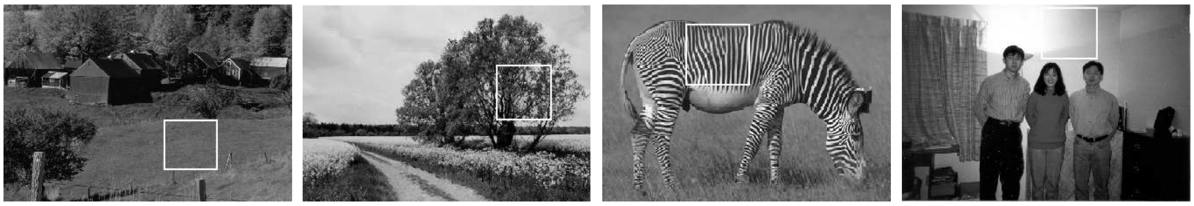
$$p(W) \propto p(K) \prod_{i=1}^{K} p(R_i)p(\ell_i)p(\Theta_i|\ell_i) \propto \exp\{-\lambda_0 K - \sum_{i=1}^{K}[\mu \oint_{\partial R_i} ds + \gamma|R_i|^c + \nu|\Theta_i|]\}.$$

2.3 The Likelihood $p(\mathbf{I}|W)$ for Grey Level Images

Visual patterns in different regions are assumed to be independent stochastic processes specified by $(\Theta_i, \ell_i), i = 1, 2, ..., K$. Thus the likelihood is,[2]

$$p(\mathbf{I}|W) = \prod_{i=1}^{K} p(\mathbf{I}_{R_i}; \Theta_i, \ell_i).$$

The choice of models needs to balance model sufficiency and computational efficiency. In studying a large image set, we found that four types of regions appear most frequently in real world images. Figure 1 shows examples for the four types of regions in windows: a). flat regions with no distinct image structures, b). cluttered regions, c).regions with homogeneous textures, and d). inhomogeneous regions with globally smooth shading variations.



(a) uniform (b) clutter (c) texture (d) shading

Figure 1 Four types of regions in the windows are typical in real world images.

We adopt the following four families of models for the four types of regions. The algorithm can switch between them by Markov chain jumps. The four families are indexed by $\ell \in \{g_1, g_2, g_3, g_4\}$ and denoted by ϖ_{g_1}, ϖ_{g_2}, ϖ_{g_3}, and ϖ_{g_4} respectively. Let $G(0; \sigma^2)$ be a Gaussian density centered at 0 with variance σ^2.

1. Grey image model family $\ell = g_1$: ϖ_{g_1}. This assumes that pixel intensities in a region R are subject to independently and identically distributed (iid) Gaussian distribution,

$$p(\mathbf{I}_R; \Theta, g_1) = \prod_{v \in R} G(\mathbf{I}_v - \mu; \sigma^2), \quad \Theta = (\mu, \sigma) \in \varpi_{g_1}. \tag{3}$$

[2]As a slight notation complication, Θ, ℓ could be viewed as parameters or hidden variables in W. We use $p(\mathbf{I}; \Theta, \ell)$ in both situations for simplicity.

2. Grey image model family $\ell = g_2$: ϖ_{g_2}. This is a non-parametric intensity histogram $h()$. In practice $h()$ is discretized as a step function expressed by a vector $(h_0, h_1, ..., h_G)$. Let n_j be the number of pixels in R with intensity level j.

$$p(\mathbf{I}_R; \Theta, g_2) = \prod_{v \in R} h(\mathbf{I}_v) = \prod_{j=0}^{G} h_j^{n_j}, \quad \Theta = (h_0, h_1, ..., h_G) \in \varpi_{g_2}. \tag{4}$$

3. Grey image model family $\ell = g_3$: ϖ_{g_3}. This is a texture model FRAME[30] with pixel interactions captured by a set of Gabor filters. This family of models was demonstrated to be sufficient in realizing a wide variety of texture patterns. To facilitate the computation, we choose a set of 8 filters and formulate the model in pseudo-likelihood form[31]. The model is specified by a long vector $\Theta = (\beta_1, \beta_2, ..., \beta_m) \in \varpi_{g_3}$, m is the total number of bins in the histograms of the 8 Gabor filtered images. Let ∂v denote the Markov neighborhood of $v \in R$, and $\mathbf{h}(\mathbf{I}_v|\mathbf{I}_{\partial v})$ the vector including 8 local histograms of filter responses in the neighborhood of pixel v. Each of the filter histogram counts the filter responses at pixels whose filter windows cover v. Thus we have

$$p(\mathbf{I}_R; \Theta, g_3) = \prod_{v \in R} p(\mathbf{I}_v|\mathbf{I}_{\partial v}; \Theta) = \prod_{v \in R} \frac{1}{Z_v} \exp\{- < \Theta, \mathbf{h}(\mathbf{I}_v|\mathbf{I}_{\partial v}) >\}, \tag{5}$$

$< \cdot, \cdot >$ is the inner product between two vectors, and the model is considered non-parametric. The reason for choosing the pseudo-likelihood expression is obvious: its normalizing constant can be computed exactly and Θ can be estimated easily from images. We refer to a recent paper[31] for discussions on the computation of this model and its variations, such as patch likelihood, etc.

4. Grey image model family g_4: ϖ_{g_4}. The first three families of models are homogeneous, which fail in characterizing regions with shading effects, such as sky, lake, wall, perspective texture, etc. In the literature, such smooth regions are often modeled by low order Markov random fields, which again do not model the inhomogeneous pattern over space and often lead to over-segmentation. In our experiments, we adopt a 2D Bezier-spline model with sixteen equally spaced control points on Λ (i.e. we fix the knots). This is a generative type model. Let $B(x, y)$ be the Bezier surface, for any $v = (x, y) \in \Lambda$,

$$B(x, y) = U_{(x)}^T \times M \times U_{(y)}, \tag{6}$$

where $U_{(x)} = ((1-x)^3, 3x(1-x)^2, 3x^2(1-x), x^3))^T$ and $M = (m_{11}, m_{12}, m_{13}, m_{14}; ...; m_{41}, ..., m_{44})$. Therefore, the image model for a region R is,

$$p(\mathbf{I}_R; \Theta, g_4) = \prod_{v \in R} G(\mathbf{I}_v - B_v; \sigma^2), \quad \Theta = (M, \sigma) \in \varpi_{g_4}. \tag{7}$$

In summary, four types of models compete to explain a grey intensity region. Whoever fits the region better will have a higher likelihood. We denote by ϖ_Θ^g the grey level model space,

$$\Theta \in \varpi_\Theta^g = \varpi_{g_1} \cup \varpi_{g_2} \cup \varpi_{g_3} \cup \varpi_{g_4}.$$

2.4 Model calibration

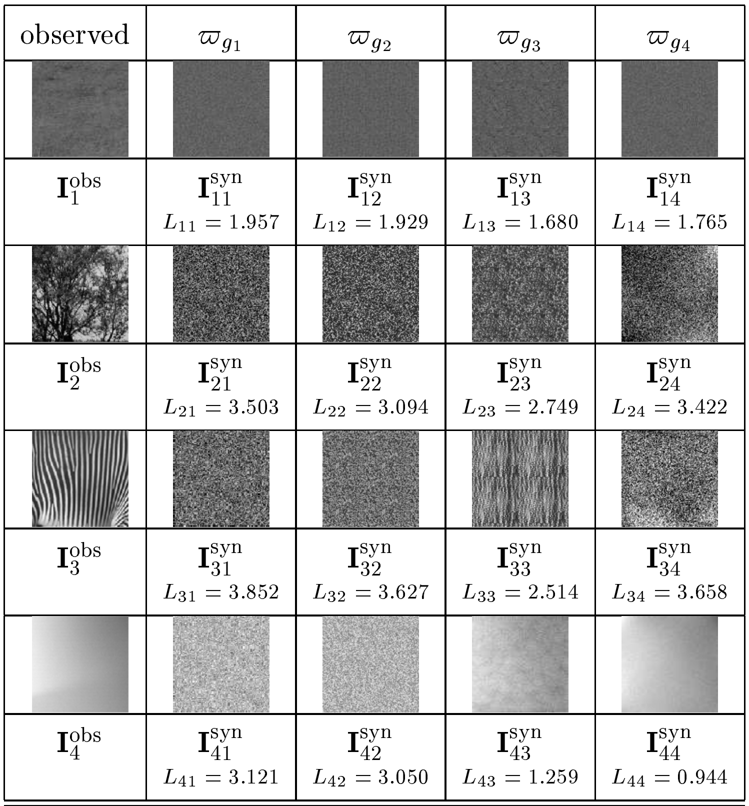


Figure 2 Comparison study of four families of models. The first column is the original image regions cropped from four real world images shown in figure 1. The images in the 2-5 columns are synthesized images $\mathbf{I}_{ij}^{\rm syn} \sim p(\mathbf{I}_R; \Theta_{ij}^*)$ sampled from the four families respectively each after an MLE fitting. The number below each synthesized image shows the per-pixel coding bits L_{ij} using each family of model.

The four image models should be calibrated for two reasons. Firstly, for computational efficiency, we prefer simple models with less parameters. However, penalizing the number of parameters is not enough in practice. When a region is of size over ~ 100 pixels, the data term dominates the prior and demands more complex models. Secondly, the pseudo-likelihood models in family ϖ_{g_3} are not a true likelihood as they depend on a rather big neighborhood, thus they are not directly comparable to the other three types of models.

To calibrate the likelihood probabilities, we did an empirical study. We collected a set of typical regions from natural images and manually divided them into four categories. For example, Figure 2 shows four typical images in the first column, which are cropped from the images in Figure 1. We denote the four images by $\mathbf{I}_i^{\text{obs}}, i = 1, 2, 3, 4$ on a lattice Λ_o. For each image $\mathbf{I}_i^{\text{obs}}$, we compute its per pixel coding length (minus log-likelihood) according to an optimal model within family ϖ_{g_j} computed by a maximum likelihood estimation for $j = 1, 2, 3, 4$.

$$L_{ij} = \min_{\varpi_{gj} \ni \Theta} -\frac{\log p(\mathbf{I}_i^{\text{obs}}; \Theta, g_j)}{|\Lambda_o|}, \quad \text{for } 1 \leq i, j \leq 4. \tag{8}$$

We denote by $\Theta_{ij}^* \in \varpi_{gj}$ optimal fit within each family and draw a typical sample (synthesis) from each fitted model,

$$\mathbf{I}_{ij}^{\text{syn}} \sim p(\mathbf{I}; \Theta_{ij}^*, g_j), \quad \text{for } 1 \leq i, j \leq 4.$$

We show $\mathbf{I}_i^{\text{obs}}$, $\mathbf{I}_{ij}^{\text{syn}}$, and L_{ij} in Figure 2 for $1 \leq i, j \leq 4$.

The results in Figure 2 show that the spline model has obviously the shortest coding length for the shading region, while the texture model fits the best for the three other regions. Then we choose to rectify these models by a constant factor e^{-c_j} for each pixel v,

$$\hat{p}(\mathbf{I}_v; \Theta, g_j) = p(\mathbf{I}_v; \Theta, g_j) e^{-c_j} \quad \text{for } j = 1, 2, 3, 4.$$

$c_j, j = 1, 2, 3, 4$ are chosen so that the rectified coding length $\hat{L}_{ij}$ reaches minimum when $i = j$. I.e. uniform regions, clutter regions, texture regions, and shading regions are best fitted by the models in ϖ_1, ϖ_2, ϖ_3, and ϖ_4 respectively.

2.5 Image models for color

In experiments, we work on both grey level and color images. For color images, we adopt a (L^*, u^*, v^*) color space and adopted three families of models indexed by $\ell \in \{c_1, c_2, c_3\}$. Let $G(0; \Sigma)$ denote a 3D Gaussian density.

1. Color image model family c_1: ϖ_{c_1}. This is an iid Gaussian model in (L^*, u^*, v^*) space.

$$p(\mathbf{I}_R; \Theta, c_1) = \prod_{v \in R} G(\mathbf{I}_v - \boldsymbol{\mu}; \Sigma), \quad \Theta = (\boldsymbol{\mu}, \Sigma) \in \varpi_{c_1}. \tag{9}$$

2. Color image model family c_2: ϖ_{c_2}. This is a mixture of two Gaussians and is used for modeling textured color regions,

$$p(\mathbf{I}_R; \Theta, c_2) = \prod_{v \in R} [\alpha_1 G(\mathbf{I}_v - \boldsymbol{\mu}_1; \Sigma_1) + \alpha_2 G(\mathbf{I}_v - \boldsymbol{\mu}_2; \Sigma_2)].$$

Thus $\Theta = (\alpha_1, \boldsymbol{\mu}_1, \Sigma_1, \alpha_2, \boldsymbol{\mu}_2, \Sigma_2) \in \varpi_{c_2}$ are the parameters.

3. Color image model family c_3: ϖ_{c_3}. We use three Bezier spline surfaces (see equation (6)) for L^*, u^*, and v^* respectively to characterize regions with gradually changing colors such as sky, wall, etc. Let $\mathbf{B}(x, y)$ be the color value in (L^*, u^*, v^*) space for any $v = (x, y) \in \Lambda$,

$$\mathbf{B}(x, y) = (U_{(x)}^T \times M_L \times U_{(y)}, U_{(x)}^T \times M_u \times U_{(y)}, U_{(x)}^T \times M_v \times U_{(y)})^T.$$

Thus the model is

$$p(\mathbf{I}_R; \Theta, c_3) = \prod_{v \in R} G(\mathbf{I}_v - \mathbf{B}_v; \Sigma),$$

where $\Theta = (M_L, M_u, M_v, \Sigma)$ are the parameters.

In summary, three types of models compete to explain a color region. Whoever fits the region better will have higher likelihood. We denote by ϖ_Θ^c the color model space, then

$$\varpi_\Theta^c = \varpi_{c_1} \cup \varpi_{c_2} \cup \varpi_{c_3}.$$

3 Anatomy of Solution Space

Before we design an algorithm, we need to study the structures of the solution space Ω in which the posterior probability $p(W|\mathbf{I})$ is distributed.

We start with the *partition space* for all possible partitions of a lattice Λ. When a lattice Λ is segmented into k disjoint regions, we call it a *k-partition* denoted by π_k,

$$\pi_k = (R_1, R_2, ..., R_k), \quad \cup_{i=1}^k R_i = \Lambda, \quad R_i \cap R_j = \emptyset, \ \forall i \neq j. \tag{10}$$

If all pixels in each region are connected, then π_k is a connected component partition[28]. The set of all k-partitions, denoted by ϖ_{π_k}, is a quotient space of the set of all possible k-colorings divided by a permutation group $\mathcal{PG}$ for the labels.

$$\varpi_{\pi_k} = \{(R_1, R_2, ..., R_k) = \pi_k; \quad |R_i| > 0, \ \forall i = 1, 2, ..., k\}/\mathcal{PG}. \tag{11}$$

Thus we have a general partition space ϖ_π with the number of regions $1 \le k \le |\Lambda|$,

$$\varpi_\pi = \cup_{k=1}^{|\Lambda|} \varpi_{\pi_k}.$$

Then the solution space for W is a union of subspaces Ω_k, and each Ω_k is a product of one k-partition space ϖ_{π_k} and k spaces for the image models

$$\Omega = \cup_{k=1}^{|\Lambda|} \Omega_k = \cup_{k=1}^{|\Lambda|} [\ \varpi_{\pi_k} \times \underbrace{\varpi_\Theta \times \cdots \times \varpi_\Theta}_{k}\ ], \tag{12}$$

where $\varpi_\Theta = \cup_{i=1}^{4} \varpi_{gi}$ for grey level images, and $\varpi_\Theta = \cup_{i=1}^{3} \varpi_{ci}$ for color images.

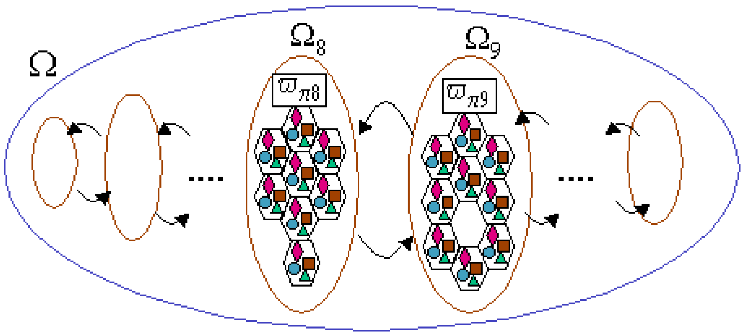


Figure 3 The anatomy of the solution space. The arrows represent Markov chain jumps, and the reversible jumps between two subspace Ω_8 and Ω_9 realize a split-and-merge of a region.

Figure 3 illustrates the structures of the solution space. In Figure 3, the four image families $\varpi_\ell, \ell = g_1, g_2, g_3, g_4$ are represented by the triangles, squares, diamonds and circles respectively. $\varpi_\Theta = \varpi_\Theta^g$ is represented by a hexagon containing the four shapes. The partition space ϖ_{π_k} is represented by a rectangle. Each subspace Ω_k consists of a rectangle and k hexagons, and each point $W \in \Omega_k$ represents a k-partition plus k image models for k regions.

We call Ω_k the **scene spaces**. ϖ_{π_k} and $\varpi_\ell, \ell = g_1, g_2, g_3, g_4$ (or $\ell = c_1, c_2, c_3$) are the basic components for constructing Ω and thus are called the **atomic spaces**. Sometimes we call ϖ_π a **partition space** and $\varpi_\ell, \ell = g_1, g_2, g_3, g_4, c_1, c_2, c_3$ the **cue spaces**.

4 Exploring the Solution Space by Ergodic Markov chains

The solution space in Figure 3 is typical for vision problems. The posterior probability $p(W|\mathbf{I})$ not only has an enormous number of local maxima but is distributed over subspaces of varying dimensions. To search for globally optimal solutions in such spaces, we adopt the Markov chain Monte Carlo (MCMC) techniques.

4.1 Three basic criteria for Markov chain design.

There are three basic requirements for Markov chain design.

Firstly, the Markov chain should be ergodic. That is, from an arbitrary initial segmentation $W_o \in \Omega$, the Markov chain can visit any other states $W \in \Omega$ in finite time. This disqualifies all greedy algorithms. Ergodicity is ensured by the jump-diffusion dynamics[12]. *Diffusion* realizes random moves within a subspace of fixed dimensions. *Jumps* realize reversible random walks between subspaces of different dimensions, as shown by the arrows in Figure 3.

Secondly, the Markov chain should be aperiodic. This is ensured by using the dynamics at random.

Thirdly, the Markov chain has stationary probability $p(W|\mathbf{I})$. This is replaced by a stronger condition of *detailed balance equations* which demands that every move should be reversible[12, 11]. The jumps in this paper all satisfy detailed balance and reversibility.

4.2 Five Markov chain dynamics

We adopt five types of Markov chain dynamics which are used at random with probabilities $p(1), ..., p(5)$ respectively. The dynamics 1-2 are diffusion, and dynamics 3-5 are reversible jumps.

Dynamics 1: boundary diffusion/competition. For mathematical convenience, we switch to a continuous boundary representation for regions $R_i, i = 1, ..., K$. These curves evolve to maximize the posterior probability through a region competition equation[29]. Let Γ_{ij} be the boundary between R_i, R_j, $\forall i, j$, and Θ_i, Θ_j the models for the two regions respectively. The motion of points $\Gamma_{ij}(s) = (x(s), y(s))$ follows the steepest ascent equation of the $\log p(W|\mathbf{I})$ plus a Brownian motion dB along the curve normal direction $\vec{n}(s)$. By variational calculus, this is[29],

$$\frac{d\Gamma_{ij}(s)}{dt} = [f_{prior}(s) + \log \frac{p(\mathbf{I}(x(s), y(s)); \Theta_i, \ell_i)}{p(\mathbf{I}(x(s), y(s)); \Theta_j, \ell_j)} + \sqrt{2T(t)}dB]\ \vec{n}(s).$$

The first two terms are derived from the prior and likelihood respectively. The Brownian motion is a normal distribution whose magnitude is controlled by a temperature $T(t)$ which decreases with time t. The Brownian motion helps to avoid local small pitfalls. The log-likelihood ratio requires that the image models are comparable. Dynamics 1 realizes diffusion within the atomic (or partition) space ϖ_{π_k} (i.e. moving within a rectangle of Figure 3).

Dynamics 2: model adaptation. This is simply to fit the parameters of a region by steepest ascent. One can add a Brownian motion, but it does not make much a difference in practice.

$$\frac{d\Theta_i}{dt} = \frac{\partial \log p(\mathbf{I}_{R_i}; \Theta_i, \ell_i)}{\partial \Theta_i}.$$

This realizes diffusion in the atomic (or cue) spaces $\varpi_l, \ell \in \{g_1, g_2, g_3, g_4, c_1, c_2, c_3\}$ (move within a triangle, square, diamond, or circle of Figure 3).

Dynamics 3-4: split and merge. Suppose at a certain time step, a region R_k with model Θ_k is split into two regions R_i and R_j with models Θ_i, Θ_j, or vice verse, and this realizes a jump between two states W to W' as shown by the arrows in Figure 3.

$$W = (K,\ (R_k, \ell_k, \Theta_k),\ W_-) \longleftrightarrow (K+1,\ (R_i, \ell_i, \Theta_i), (R_j, \ell_j, \Theta_j),\ W_-) = W',$$

where W_- denotes the remaining variables that are unchanged during the move. By the classic Metropolis-Hastings method[19], we need two proposal probabilities $G(W \to dW')$ and $G(W' \to dW)$. $G(W \to dW')$ is a conditional probability for how likely the Markov chain *proposes* to move to W' at state W, and $G(W' \to dW)$ is the proposal probability for coming back. The proposed split is then accepted with probability

$$\alpha(W \to dW') = \min(1, \frac{G(W' \to dW)p(W'|\mathbf{I})dW'}{G(W \to dW')p(W|\mathbf{I})dW}).$$

There are two routes (or "pathways" in a psychology language) for computing the split proposal $G(W \to dW')$.

In route 1, it first chooses a split move with probability $q(3)$, then chooses region R_k from a total of K regions at random, we denote this probability by $q(R_k)$. Given R_k, it chooses a candidate splitting boundary Γ_{ij} within R_k with probability $q(\Gamma_{ij}|R_k)$. Then for the two new regions R_i, R_j it chooses two new model types ℓ_i and ℓ_j with probabilities $q(\ell_i)$ and $q(\ell_j)$ respectively. Then it chooses $\Theta_i \in \varpi_{\ell_i}$ with probability $q(\Theta_i|R_i, \ell_i)$ and chooses Θ_j with probability $q(\Theta_j|R_j, \ell_j)$. Thus,

$$G(W \to dW') = q(3)q(R_k)q(\Gamma_{ij}|R_k)q(\ell_i)q(\Theta_i|R_i, \ell_i)q(\ell_j)q(\Theta_j|R_j, \ell_j)dW'. \tag{13}$$

In route 2, it first chooses two new region models Θ_i and Θ_j, and then decides the boundary Γ_{ij}. Thus,

$$G(W \to dW') = q(3)q(R_k)q(\ell_i)q(\ell_j)q(\Theta_i, \Theta_j|R_k, \ell_i, \ell_j)q(\Gamma_{ij}|R_k, \Theta_i, \Theta_j)dW'. \tag{14}$$

We shall discuss in later subsection that either of the two routes can be more effective than the other depending on the region R_k.

Similarly we have the merge proposal probability,

$$G(W' \to dW) = q(4)q(R_i, R_j)q(\ell_k)q(\Theta_k|R_k, \ell_k)dW. \tag{15}$$

$q(R_i, R_j)$ is the probability of choosing to merge two regions R_i and R_j at random.

Dynamics 5: switching image models. This switches the image model within the four families (three for color images) for a region R_i. For example, from texture description to a spline surface etc.

$$W = (\ell_i, \Theta_i, W_-) \longleftrightarrow (\ell'_i, \Theta'_i, W_-) = W'.$$

The proposal probabilities are

$$G(W \to dW') = q(5)q(R_i)q(\ell'_i)q(\Theta'_i|R_i, \ell'_i)dW', \tag{16}$$

$$G(W' \to dW) = q(5)q(R_i)q(\ell_i)q(\Theta_i|R_i, \ell_i)dW. \tag{17}$$

4.3 The bottlenecks

The speed of a Markov chain depends critically on the design of its proposal probabilities in the jumps. In our experiments, the proposal probabilities, such as $q(1), ..., q(5)$, $q(R_k)$, $q(R_i, R_j)$, $q(\ell)$ are easy to specify and do not influence the convergence significantly. The real bottlenecks are caused by two proposal probabilities in the jump dynamics.

1. $q(\Gamma|R)$ in eqn.(13): Where is a good Γ for splitting a given region R? $q(\Gamma|R)$ is a probability in the atomic (or partition) space ϖ_π.

2. $q(\Theta|R, \ell)$ in eqns (13), (15) and (17): For a given region R and a model family $\ell \in \{g_1, ..., g_4, c_1, c_2, c_3\}$, what is a good Θ? $q(\Theta|R, \ell)$ is a probability in the atomic (cue) space ϖ_ℓ.

It is worth mentioning that both probabilities $q(\Gamma|R)$ and $q(\Theta|R, \ell)$ cannot be replaced by deterministic decisions which were used in region competition[29] and others[15]. Otherwise, the Markov chain will not be reversible and thus reduce to a greedy algorithm. On the other hand, if we choose uniform distributions, it is equivalent to blind search, and the Markov chain will experience exponential "waiting" time before each jump. In fact, the length of the waiting time is proportional to the volume of the cue spaces. The design of these probabilities need to strike a balance between *speed* and *robustness* (non-greediness).

While it is hard to analytically derive a convergence rate for complicated algorithms that we are dealing with, it is revealing to observe the following theorem in a simple case[18]

Theorem 1 *Sampling a target density $p(x)$ by independence Metropolis-Hastings algorithm with proposal probability $q(x)$. Let $P^n(x_o, y)$ be the probability of a random walk to reach point y at n steps. If there exists $\rho > 0$ such that,*

$$\frac{q(x)}{p(x)} \geq \rho, \quad \forall x,$$

then the convergence measured by a L_1 norm distance

$$||P^n(x_o, \cdot) - p|| \leq (1 - \rho)^n.$$

This theorem states that the proposal probability $q(x)$ should be very close to $p(x)$ for fast convergence. In our case, $q(\Gamma|R)$ and $q(\Theta|R,\ell)$ should be equal to the conditional probabilities of some marginal probabilities of the posterior $p(W|\mathbf{I})$ within the atomic spaces ϖ_π and ϖ_ℓ respectively. That is,

$$q^*_\Gamma(\Gamma_{ij}|R_k) = p(\Gamma_{ij}|\mathbf{I}, R_k), \quad q^*_\Theta(\Theta|R,\ell) = p(\Theta|\mathbf{I}, R, \ell), \ \ \forall \ell. \tag{18}$$

Unfortunately, q^*_Γ and q^*_Θ have to integrate information from the entire image $\mathbf{I}$, and thus are intractable. We must seek approximations, and this is where the data-driven methods step in.

In the next section, we discuss data clustering for each atomic space ϖ_ℓ, $\ell \in \{c_1, c_2, c_3\}$ and $\ell \in \{g_1, g_2, g_3, g_4\}$ and edge detection in ϖ_π. The results of clustering and edge detection are expressed as non-parametric probabilities for approximating the ideal marginal probabilities q^*_Γ and q^*_Θ in these atomic space respectively.

5 Data-Driven Methods

5.1 Method I: clustering in atomic spaces ϖ_ℓ for $\ell \in \{c_1, c_2, c_3, g_1, g_2, g_3, g_4\}$

Given an image $\mathbf{I}$ (grey or color) on lattice Λ, we extract a feature vector F^ℓ_v at each pixel $v \in \Lambda$. The dimension of F^ℓ_v depends on the image model indexed by ℓ. Then we have a collection of vectors

$$\mho^\ell = \{F^\ell_v : \ v \in \Lambda\}.$$

In practice, v can be subsampled for computational ease. The set of vectors are clustered by either an EM method[7] or a mean-shift clustering[5, 6] algorithm to $\mho^\ell$. The EM-clustering approximates the points density in $\mho^\ell$ by a mixture of m Gaussians, and it extends from the m-mean clustering by a soft cluster assignment to each vector F_v. The mean-shift algorithm assumes a non-parametric distribution for $\mho^\ell$ and seeks the modes (local maxima) in its density (after some Gaussian window smoothing). Both algorithms return a list of m weighted clusters $\Theta^\ell_1, \Theta^\ell_2, ..., \Theta^\ell_m$ with weights $\omega^\ell_i, i = 1, 2, ..., m$, and we denote by

$$\mathcal{P}^\ell = \{ \ (\omega^\ell_i, \Theta^\ell_i) : \ \ i = 1, 2, ..., m. \ \}. \tag{19}$$

We call $(\omega_i^\ell, \Theta_i^\ell)$ a *weighted atomic (or cue) particle* in ϖ_ℓ for $\ell \in \{c_1, c_3, g_1, g_2, g_3, g_4\}$. [3] The size m is chosen to be conservative, or it can be computed in a coarse-to-fine strategy with a limit $m = |\mho^\ell|$. This is well discussed in the literature[5, 6].

In the clustering algorithms, each feature F_v^ℓ and thus its location v is classified to a cluster Θ_i^ℓ with probability $S_{i,v}^\ell$,

$$S_{i,v}^\ell = p(F_v^\ell; \Theta_i^\ell), \qquad \text{with} \sum_{i=1}^{m} S_{i,v}^\ell = 1, \quad \forall v \in \Lambda, \ \forall \ell.$$

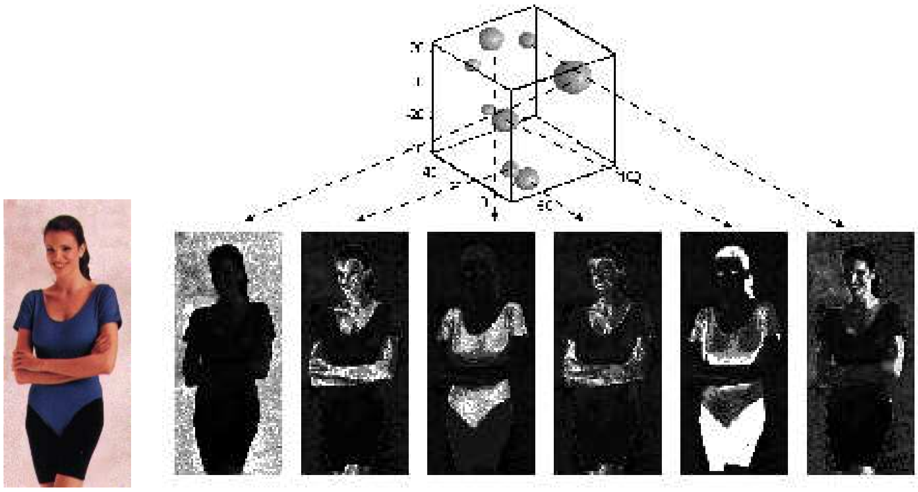
This is a soft assignment and can be computed by the distance from F_v to the cluster centers. We call

$$S_i^\ell = \{S_{i,v}^\ell : \ v \in \Lambda\}, \quad \text{for } \ i = 1, 2, ..., m, \ \forall \ell \tag{20}$$

a **saliency map** associated with cue particle Θ_i^ℓ.

In the following, we discuss each model family with experiments.

5.1.1 *Computing cue particles in ϖ_{c_1}*



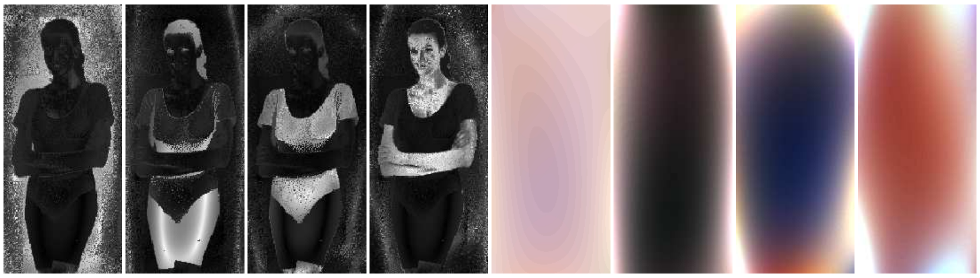
Input **I** Color clusters and their saliency maps $S_i^{c_1}, i = 1, ..., 6$

Figure 4 A color image and its clusters in (L^*, u^*, v^*) space for ϖ_{c_1}, the second row are six of the saliency maps associated with the color clusters.

[3] The atomic space ϖ_{c_2} is a composition of two ϖ_{c_1}, and thus is computed from ϖ_{c_1}.

For color images, we take $F_v = (L_v, U_v, V_v)$ and apply a mean-shift algorithm[5, 6] to compute color clusters in ϖ_{c_1}. For example, Figure 4 shows a few color clusters (balls) in a cubic ((L^*, u^*, v^*)-space) for a simple color image (left), the size of the balls represents the weights $\omega_i^{c_1}$. Each cluster is associated with a saliency map $S_i^{c_1}$ for $i = 1, 2, ..., 6$ in the second row, and the bright areas mean high probabilities. From left to right are respectively, background, skin, shirt, shadowed skin, pant and hair, highlighted skin.

5.1.2 Computing cue particles in ϖ_{c3}



(a) (b) (c) (d) (e) (f) (g) (h)

Figure 5 (a)-(d) are saliency maps associated with four clusters in ϖ_{c_3}. (e)-(h) are the color spline surfaces for the four clusters.

Each point v contributes its color $\mathbf{I}_v = (L_v, U_v, V_v)$ as "surface heights", and we apply an EM-clustering to find the spline surface models. Figure 5 shows the clustering result for the woman image. Figures 5.a-d are saliency maps $S_i^{c_3}$ for $i = 1, 2, 3, 4$. Figures 5.e-h are the four reconstructed images according to fitted spline surfaces which recover some global illumination variations.

5.1.3 Computing cue particles in ϖ_{g_1}

In this model, the feature space $F_v = \mathbf{I}_v$ is simply the intensity, and $\mho^{g_1}$ is the image intensity histogram. We simply apply a mean-shift algorithm to get the modes (peaks) of the histogram and the breadth of each peak decides its variance.

Figure 6 shows six saliency maps $S_i^{g_1}, i = 1, 2, ..., 6$ for a zebra image (the original image is shown in Figure 14.a. In the clustering map on the left in Figure 6, each pixel is assigned to its most likely particle.

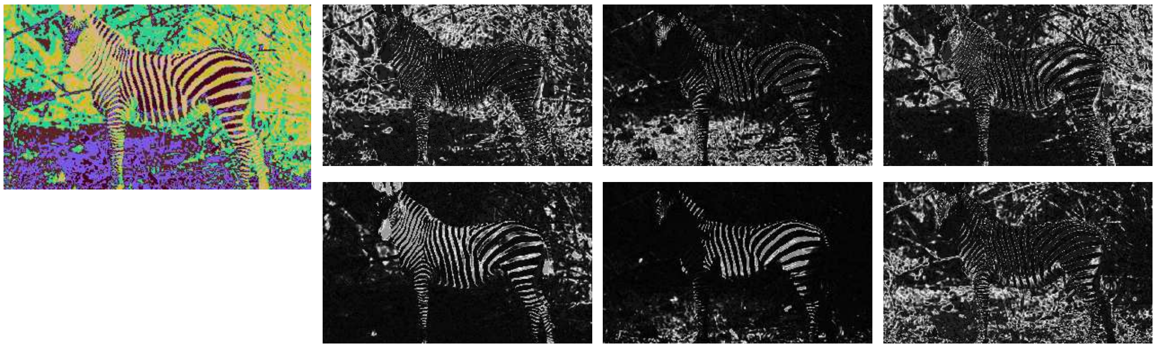


Figure 6 A clustering map (left) for ϖ_{g_1} and six saliency maps $S_i^{g_1}, i = 1, ..., 6$ of a zebra image (input is in Fig. 14.a).

5.1.4 Computing the cue particles in ϖ_{g2}

For clustering in ϖ_{g2}, at each subsampled pixel $v \in \Lambda$, we compute F_v as a local intensity histogram $F_v = (h_{v0}, ..., h_{vG})$ accumulated over a local window centered at v. Then an EM clustering is applied to compute the cue particles, and each particle $\Theta_i^{g_2}, i = 1, ..., m$ is a histogram. This model is used for clutter regions.

Figure 7 shows the clustering results on the same zebra image.

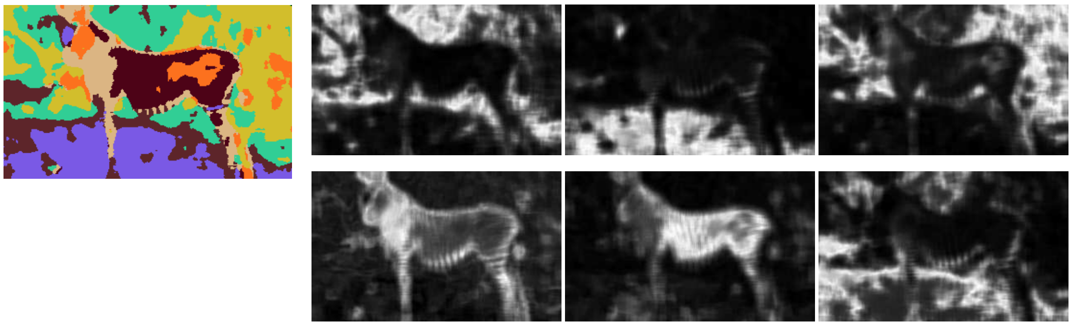


Figure 7 A clustering map (left) for ϖ_{g_2} and six saliency maps $S_i^{g_2}, i = 1..6$ of a zebra image (input is inFig. 14.a).

5.1.5 *Computing cue particles* ϖ_{g_3}

At each subsampled pixel $v \in \Lambda$, we compute a set of 8 local histograms for 8 filters over a local window of 12×12 pixels. We choose 8 filters for computational convenience: one δ filter, two gradient filters, one Laplacian of Gaussian filter, and four Gabor filters. Each histogram has 9 bins. Then $F_v^{g_3} = (h_{v,1,1}, ..., h_{v,8,9})$ is the feature. An EM clustering is applied to find the m mean histograms $\bar{\mathbf{h}}_i, i = 1, 2, ..., m$. We can compute the cue particles for texture models $\Theta_i^{g_3}$ from $\bar{\mathbf{h}}_i$ for $i = 1, 2, ..., m$. A detailed account of this transform is referred to a previous paper[31].

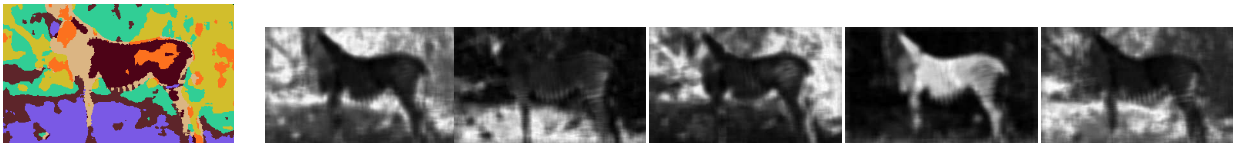


Figure 8 Texture clustering. A clustering map (left) and five saliency maps for five particles $\Theta_i^{g_3}, i = 1, 2, ..., 5$.

Figure 8 shows the texture clustering results on the zebra image with one clustering map on the left, and five saliency maps for five particles $\Theta_i^{g_3}, i = 1, 2, ..., 5$.

5.1.6 *Computing cue particles in* ϖ_{g_4}

Each point v contributes its intensity $\mathbf{I}_v = F_v$ as a "surface height", and we apply an EM-clustering to find the spline surface models. Figure 9 shows a clustering result for the zebra image with four surfaces. The second row shows the four surfaces which recover some global illumination variations. Unlike the texture clustering results which capture the zebra strips as a whole region, the surface models separate the black and white stripes as two regions – another valid perception. Interestingly, the black and white strips in the zebra skin both have shading changes which are fitted by the spline models.

5.2 *Method II: Edge detection*

We detect intensity edges using Canny edge detector[4] and color edges using a method in [17], and trace edges to form a partition of the image lattice. We choose edges at three

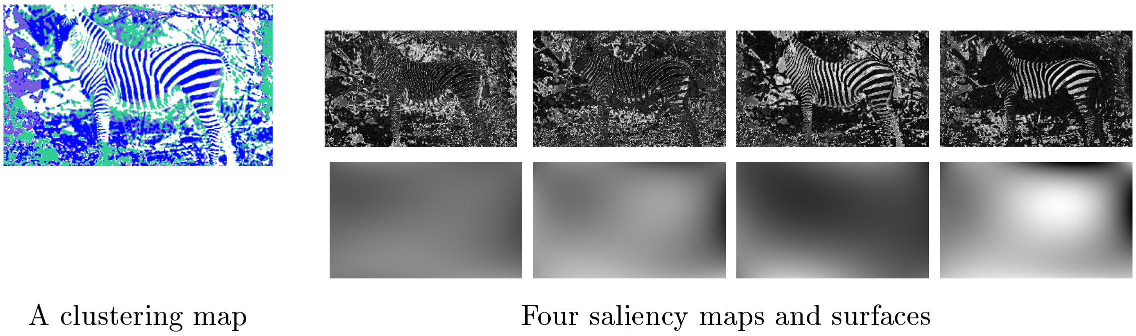


Figure 9 Clustering result on the zebra image under Bezier surface model. The left image is the clustering map. The first row of images on the right side are the saliency maps. The second row shows the fitted surfaces using the surface height as intensity.

scales according to edge strength, and thus compute the partition maps in three coarse-to-fine scales. We choose not to discuss the details, but show some results using the two running examples: the woman and zebra images.

Figure 10.a shows a color image and three scales of partitions. Since this image has strong color cue, the edge maps are very informative about where the region boundaries are. In contrast, the edge maps for the zebra image are very messy, as Figure 11 shows.

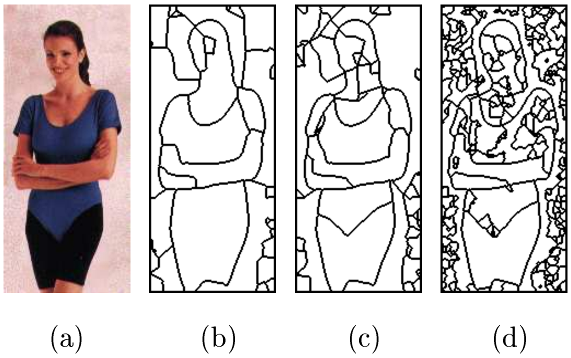


Figure 10 Partition maps at three scales of details for a color image. (a) Input image. (b) Partition map at scale 1. (c) Partition map at scale 2. (d) Partition map at scale 3.

(a) (b) (c) (d)

Figure 11 A grey level image and three partition maps at three scales. (a) Input image. (b) Partition map at scale 1. (c) Partition map at scale 2. (d) Partition map at scale 3.

6 Computing importance proposal probabilities

It is generally acknowledged in the community that clustering and edge detection algorithms can sometimes produce good segmentations or even perfect results for some images, but very often they are far from being reliable for generic images, as the experiments in Figures 4-11 demonstrate. It is also true that sometimes one of the image models and edge detection scales could do a better job in segmenting some regions than other models and scales, but we do not know a priori what types of regions present in a generic image. Thus we compute all models and edge detection at multiple scales and then utilize the clustering and edge detection results probabilistically. MCMC theory provides a framework for integrating these probabilistic information in a principled way under the guidance of a globally defined Bayesian posterior probability.

We explain how the the importance proposal probabilities $q(\Theta|R,\ell)$ and $q(\Gamma_{ij}|R_k)$ in Section 4.3. are computed from the data-driven results.

6.1 Computing importance proposal probability $q(\Theta|R,\ell)$

The clustering method in an atomic (cue) space ϖ_ℓ outputs a set of weighted cue particles $\mathcal{P}^\ell$. $\mathcal{P}^\ell$ encodes a non-parametric probability in ϖ_ℓ,

$$q(\Theta|\Lambda,\ell) = \sum_{i=1}^{m} \omega_i^\ell G(\Theta - \Theta_i^\ell), \quad \text{with} \ \sum_{i=1}^{m} \omega_i^\ell = 1, \tag{21}$$

where $G(x)$ is a Parzen window centered at 0. As a matter of fact, $q(\Theta|\Lambda,\ell) = q(\Theta|\mathbf{I}))$ is an approximation to a marginal probability of the posterior $p(W|\mathbf{I})$ on cue space $\varpi_\ell, \ell \in \{g_1, g_2, g_3, g_4, c_1, c_3\}$, since the partition π is integrated out in EM-clustering.

$q(\Theta|\Lambda, \ell)$ is computed once for the whole image, and $q(\Theta|R, \ell)$ is computed from $q(\Theta|\Lambda, \ell)$ for each R at run time. It proceeds in the following. Each cluster $\Theta_i^\ell, i = 1, 2, ..., m$ receives a real-valued vote from the pixel $v \in R$ in region R, and the accumulative vote is the summation of the saliency map S_i^ℓ associated with Θ_i^ℓ, i.e.,

$$p_i = \frac{1}{|R|} \sum_{v \in R} S_{i,v}^\ell, \quad i = 1, 2, ..., m, \;\; \forall \ell.$$

Obviously the clusters which receive high votes should have high chance to be chosen. Thus we sample a new image model Θ for region R,

$$\Theta \sim q(\Theta|R, \ell) = \sum_{i=1}^{m} p_i G(\Theta - \Theta_i^\ell). \tag{22}$$

Equation (22) explains how we choose (or propose) an image model for a region R. We first draw a cluster i at random according to probability $p = (p_1, p_2, ..., p_m)$, and then do a random perturbation at Θ_i^ℓ. Thus any $\Theta \in \varpi_\ell$ has a non-zero probability to be chosen for robustness and ergodicity. Intuitively the clustering results with local votes propose the "hottest" portions of the space in a probabilistic way to guide the jump dynamics.

In practice, one could implement a multi-resolution (on a pyramid) clustering algorithm over smaller local windows, thus the clusters $\Theta_i^\ell, i = 1, 2, ..., m$ will be more effective at the expense of some overhead computing.

6.2 Computing importance proposal probability $q(\Gamma|R)$

By edge detection and tracing, we obtain partition maps denoted by $\Delta^{(s)}$ at multiple scales $s = 1, 2, 3$. In fact, each partition map $\Delta^{(s)}$ consists of a set of "meta-regions" $r_i^{(s)}, i = 1, 2, ..., n$,

$$\Delta^{(s)}(\Lambda) = \{r_i^{(s)} : \;\; i = 1, 2, ..., n, \cup_{i=1}^{n} r_i^{(s)} = \Lambda\}, \quad \text{for } s = 1, 2, 3.$$

These meta regions are then used in combination to form $K \leq n$ regions $R_1^{(s)}, R_2^{(s)}, ..., R_K^{(s)}$,

$$R_i^{(s)} = \cup_j r_j^{(s)}, \quad \text{with } r_j^{(s)} \in \Delta^{(s)}, \;\; \forall i = 1, 2, ..., K.$$

One could put a constraint that all meta-regions in a region $R_i^{(s)}$ are connected.

Let $\pi_k^{(s)} = (R_1^{(s)}, R_2^{(s)}, ..., R_k^{(s)})$ denote a k-partition based on $\Delta^{(s)}$. $\pi_k^{(s)}$ is different from the general k-partition π_k because regions $R_i^{(s)}, i = 1, ..., K$ in $\pi_k^{(s)}$ are limited to the meta-regions. We denote by $\Pi_k^{(s)}$ the set of all k-partitions based on a partition map $\Delta^{(s)}$.

$$\Pi_k^{(s)} = \{(R_1^{(s)}, R_2^{(s)}, ..., R_k^{(s)}) = \pi_k^{(s)} : \quad \cup_{i=1}^k R_i^{(s)} = \Lambda\}. \tag{23}$$

We call each $\pi_k^{(s)}$ in $\Pi_k^{(s)}$ a *k-partition particle* in atomic (partition) space ϖ_{π_k}. Like the clusters in a cue space, $\Pi_k^{(s)}$ is a sparse subset of ϖ_{π_k}, and it narrows the search in ϖ_{π_k} to the most promising portions.

So each partition map $\Delta^{(s)}$ encodes a probability in the atomic (partition) space ϖ_{π_k}.

$$q^{(s)}(\pi_k) = \frac{1}{|\Pi_k^{(s)}|} \sum_{j=1}^{|\Pi_k^{(s)}|} G(\pi_k - \pi_{k,j}^{(s)}), \quad \text{for } s = 1, 2, 3. \ \forall k. \tag{24}$$

$G()$ is a smooth window centered at 0 and its smoothness accounts for boundary deformations and forms a cluster around each partition particle, and $\pi_k - \pi_{k,j}^{(s)}$ measures the difference between two partition maps π_k and $\pi_{k,j}^{(s)}$. Martin et al.[21] recently proposed a method of measuring such difference and we use a simplified version. In the finest resolution, all meta regions reduce to pixels, and $\Pi_k^{(s)}$ is then equal to the atomic space ϖ_{π_k}. We adopt equal weights for all partitions $\pi_k^{(s)}$, and one may add other geometric preferences to some partitions.

In summary, the partition maps at all scales encode a non-parametric probability in ϖ_{π_k},

$$q(\pi_k) = \sum_s q(s) q^{(s)}(\pi_k), \quad \forall k.$$

This $q(\pi_k)$ can be considered as an approximation to the marginal posterior probability $p(\pi_k|\mathbf{I})$.

The partition maps $\Delta^{(s)}, \forall s$ (or $q(\pi_k), \forall k$ implicitly) are computed once for the whole image, then the importance proposal probability $q(\Gamma|R)$ is computed from $q(\pi_k)$ for each region as a conditional probability at run time, like in the cue spaces.

Figure 12 illustrates an example. We show partition maps $\Delta^{(s)}(\Lambda)$ at three scales, and the edges are shown at width $3, 2, 1$ respectively for $s = 1, 2, 3$. A candidate region R is proposed to split. $q(\Gamma|R)$ is the probability for proposing a splitting boundary Γ.

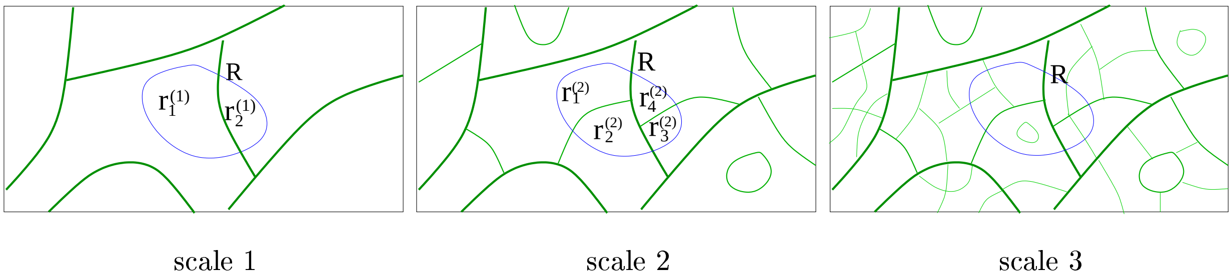


Figure 12 A candidate region R_k is superimposed on the partition maps at three scales for computing a candidate boundary Γ_{ij} for the pending split.

We superimpose R on the three partition maps. The intersections between R and the meta regions generate three sets

$$\Delta^{(s)}(R) = \{r_j^{(s)} : r_j^{(s)} = R \cap r_j \text{ for } r_j \in \Delta^{(s)}(\Lambda), \text{ and } \cup_i r_i^{(s)} = R\}, \quad s = 1, 2, 3.$$

For example, in Figure 12, $\Delta^{(1)}(R) = \{r_1^{(1)}, r_2^{(1)}\}$, $\Delta^{(2)}(R) = \{r_1^{(2)}, r_2^{(2)}, r_3^{(2)}, r_4^{(2)}\}$, and so on.

Thus we can define $\pi_c^{(s)}(R) = (R_1^{(s)}, R_2^{(s)}, ..., R_c^{(s)})$ as a c-partition of region R based on $\Delta^{(s)}(R)$, and define a c-partition space of R as

$$\Pi_c^{(s)}(R) = \{(R_1^{(s)}, R_2^{(s)}, ..., R_c^{(s)}) = \pi_c^{(s)}(R) : \quad \cup_{i=1}^{c} R_i^{(s)} = R\}, \quad \forall s. \tag{25}$$

We can define distributions on $\Pi_c^{(s)}(R)$.

$$q^{(s)}(\pi_c(R)) = \frac{1}{|\Pi_c^{(s)}(R)|} \sum_{j=1}^{|\Pi_c^{(s)}(R)|} G(\pi_c - \pi_{c,j}^{(s)}(R)), \quad \text{for } s = 1, 2, 3, \ \forall c. \tag{26}$$

Thus one can propose to split R into c pieces, in a general case,

$$\pi_c(R) \sim q(\pi_c(R)) = \sum_s q(s) q^{(s)}(\pi_c(R)).$$

That is, we first select a scale s with probability $q(s)$. $q(s)$ depends on R. For example, for a large region R we can choose coarse scale with higher probability, and choose a fine scale for small regions. Then we choose a c-partition from the set $\Pi_c^{(s)}(R)$. In our implementation, $c = 2$ is chosen as a special case for easy implementation. It is trivial to show that an arbitrary c-partition of region R, $\pi_c(R)$, can be generated through composing $\pi_2(R)$ in multiple steps. Obviously there is a big overhead for choosing large c.

6.3 Computing $q(\Theta_i, \Theta_j | R, \ell_i, \ell_j)$ *and* $q(\Gamma_{ij} | R, \Theta_i, \Theta_j)$

In some cases, we find the second route useful for splitting a region which we discussed in designing MCMC dynamics 3-4 (see equation (14)).

For example, there are two ways to perceive the zebra in Figure 14. One perceives the zebra as one textured region (by a model in ϖ_{g3}). The other sees it as one region of black stripes plus one region of white strips and thus uses two models in ϖ_{g_1} or ϖ_{g_4}. The Markov chain should be able to switch between the two perceptions effectively (see results in Figure 14.b-d). This is necessary and typical for the transitions between any texture regions and intensity regions.

Because the number of strips in such textures is large, the first split procedure (route 1) is very ineffective, and it works on one strip at a time. This motivates the second pathway for split dynamics.

For a candidate region R, we first propose two new region models (we always assume the same labels $\ell_i = \ell_j$), this can be done by twice sampling the importance proposal probabilities $q(\Theta | R, \ell)$, so

$$(\Theta_i, \Theta_j) \sim q(\Theta_i, \Theta_j | R, \ell_i, \ell_j) = q(\Theta_i | R, \ell_i) q(\Theta_j | R, \ell_j).$$

Obviously we exclude Θ_i from the candidate set when we select Θ_j. Then we decide on the boundary Γ $q(\Gamma_{ij} | R, \Theta_i, \Theta_j)$ by randomly labeling the pixels in R according to probabilities of the saliency maps.

6.4 A unifying framework

To summarize this section, the DDMCMC paradigm provides a unifying framework for understanding the roles of many existing image segmentation algorithms. Firstly, edge detection and tracing methods[4, 17] compute implicitly a marginal probability $q(\pi | \mathbf{I})$ on the partition space ϖ_π. Secondly, clustering algorithms[5, 6] compute a marginal probability on the model space ϖ_ℓ for various models ℓ. Thirdly, the split-and-merge and model switching[2] realize jump dynamics. Fourthly, region growing and competition methods[29, 24] realize diffusion dynamics for evolving the region boundaries.

7 Computing Multiple Distinct Solutions

7.1 Motivation and a mathematical principle

The DDMCMC paradigm samples solutions from the posterior $W \sim p(W|\mathbf{I})$ endlessly, as we argued in the introduction that segmentation is a computing process not a task. To extract an optimal result, one can take an annealing strategy and use the conventional *maximum a posteriori* (MAP) estimator

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

In this paper we argue that it is desirable and often critical to have the ability of computing multiple distinct solutions for the following reasons.

Firstly, natural scenes are intrinsically ambiguous, and for an image $\mathbf{I}$ many competing organizations and interpretations exist in visual perception.

Secondly, for robustness, decisions should be left to the last stage of computation when a segmentation process is integrated with a specific task. Therefore it is best to maintain a set of *typical* solutions.

Thirdly, preserving multiple solutions is necessary when the prior and likelihood models are not perfect. Because the globally optimal solution may not be semantically more meaningful than some other inferior local maxima.

However, simply keeping a set of samples from the Markov chain sequence is not enough, because it often collects a set of segmentations which are trivially different from each other. Here we present a mathematical principle for computing important and distinctive solutions in space Ω. (Our result was early presented in a CVPR2000 paper).

Let $S = \{(\omega_i, W_i) : i = 1, ..., K\}$ be a set of K weighted solutions which we call "scene particles", the weight is its posterior probability $\omega_i = p(W|\mathbf{I}), i = 1, 2, ..., K$. (Note that there is a slight abuse of notation, we use K for the number of regions in W before. Here it is a different K). S encodes a non-parametric probability in Ω,

$$\hat{p}(W|\mathbf{I}) = \sum_{i=1}^{K} \frac{\omega_i}{\boldsymbol{\omega}} G(W - W_i), \quad \sum_{i=1}^{K} \omega_i = \boldsymbol{\omega}.$$

G is a Gaussian window in Ω.

As all image ambiguities are captured in the Bayesian posterior probability, to reflect the intrinsic ambiguities, we should compute the set of solutions S which best preserves the posterior probability. Thus we let $\hat{p}(W|\mathbf{I})$ approaches $p(W|\mathbf{I})$ by minimizing a Kullback-Leibler divergence $D(p||\hat{p})$ under a complexity constraint $|S| = K$,

$$S^* = \arg\min_{|S|=K} D(p||\hat{p}) = \arg\min_{|S|=K} \int p(W|\mathbf{I}) \log \frac{p(W|\mathbf{I})}{\hat{p}(W|\mathbf{I})} dW. \quad (27)$$

This criterion extends the conventional MAP estimator (see appendix for further discussion).

7.2 A K-adventurers algorithm for multiple solutions

Fortunately, the KL-divergence $D(p||\hat{p})$ can be estimated fairly accurately by a distance measure $\hat{D}(p||\hat{p})$ which is computable, thanks to two observations of the posterior probability $p(W|\mathbf{I})$ which has many separable modes. The details of the calculation and some experiments are given in the appendix. The idea is simple. We can always represent $p(W|\mathbf{I})$ by a mixture of Gaussian, i.e. a set of N particles with N large enough. By ergodicity, the Markov chain is supposed to visit these significant modes over time! Thus our goal is to extract K distinct solutions from the Markov chain sampling process.

Here we present a greedy algorithm for computing S^* approximately. We call the algorithm – "K-adventurers algorithm. [4]

Suppose we have a set of K particles S at step t. At time $t+1$, we obtain a new particle (or a number of particles) by MCMC, usually following a successful jump. We augment the set S to S_+ by adding the new particle(s). Then we eliminate one particle (or a number of particles) from S_+ to get a new S_{new} by minimizing the approximative KL divergence. divergence $\hat{D}(p_+||p_{\text{new}})$.

The k-adventurers algorithm

1. Initializing S and $\hat{p}$ by repeating one initial solution K times.
2. Repeat
3. Compute a new particle $(\omega_{K+1}, \mathbf{x}_{K+1})$ by DDMCMC after a successful jump.
4. $S_+ \leftarrow S \bigcup \{(\omega_{K+1}, \mathbf{x}_{K+1})\}$.
5. $\hat{p} \leftarrow S_+$.
6. For $i = 1, 2, \ldots, K+1$ do
7. $S_{-i} \leftarrow S_+ / \{(\omega_i, \mathbf{x}_i)\}$.

[4] The name follows a statistics metaphor told by Mumford to one of the authors Zhu. A team of K adventurers want to occupy K largest islands in an ocean while keeping apart from each other's territories.

8. $\hat{p}_{-i} \leftarrow S_{-i}$.
9. $d_i = D(p||\hat{p}_{-i})$.
10. $i^* = \arg\min_{i \in \{1,2,\ldots,K+1\}} d_i$.
11. $S \leftarrow S_{-i^*}, \ \ \hat{p} \leftarrow \hat{p}_{-i^*}$

In practice, we run multiple Markov chains and add new particles to the set S in a batch fashion. From our experiments, the greedy algorithm did a satisfactory job and it is shown to be optimal in two low dimensional examples in the Appendix.

8 Experiments

The DDMCMC paradigm was tested extensively on many grey level, color, and textured images. This section shows some examples and more are available on our website[5]. It was also tested in a benchmark dataset of 50 natural images in both color and grey level[21] by the Berkeley group[6], where the results by DDMCMC and other methods such as [26] are displayed in comparison to those by a number of human subjects. Each tested algorithm uses the same parameter setting for all the benchmark images and thus the results were obtained purely automatically.

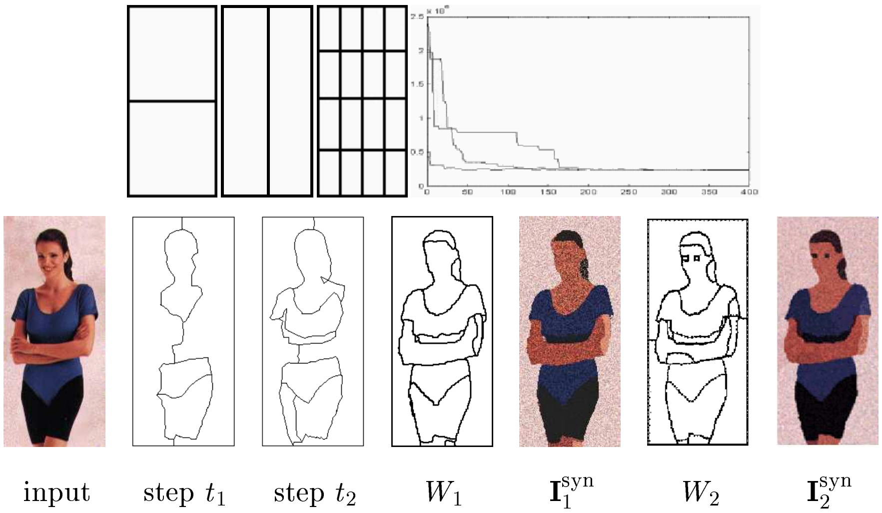


Figure 13 Segmenting a color image by DDMCMC with two solutions. See text for explanation.

[5] See www.cis.ohio-state.edu/oval/Segmentation/DDMCMC/DDMCMC.htm

[6] See www.cs.berkeley.edu/~dmartin/segbench/BSDS100/html/benchmark

We first show our working example on the color woman image. Following the importance proposal probabilities for the edges in Figure 10 and for color clustering in Figure 4, we simulated three Markov chains with three different initial segmentations shown in Figure 13 (top row). The energy changes ($-\log p(W|\mathbf{I})$) of the three MCMCs are plotted in Figure 13 against time steps. Figure 13 shows two different solutions W_1, W_2 obtained by a Markov chain using K-adventurers algorithm. To verify the computed solution W_i, we synthesized an image by sampling from the likelihood $\mathbf{I}_i^{\rm syn} \sim p(\mathbf{I}|W_i), i = 1, 2$. The synthesis is a good way to examine the sufficiency of models in segmentation.

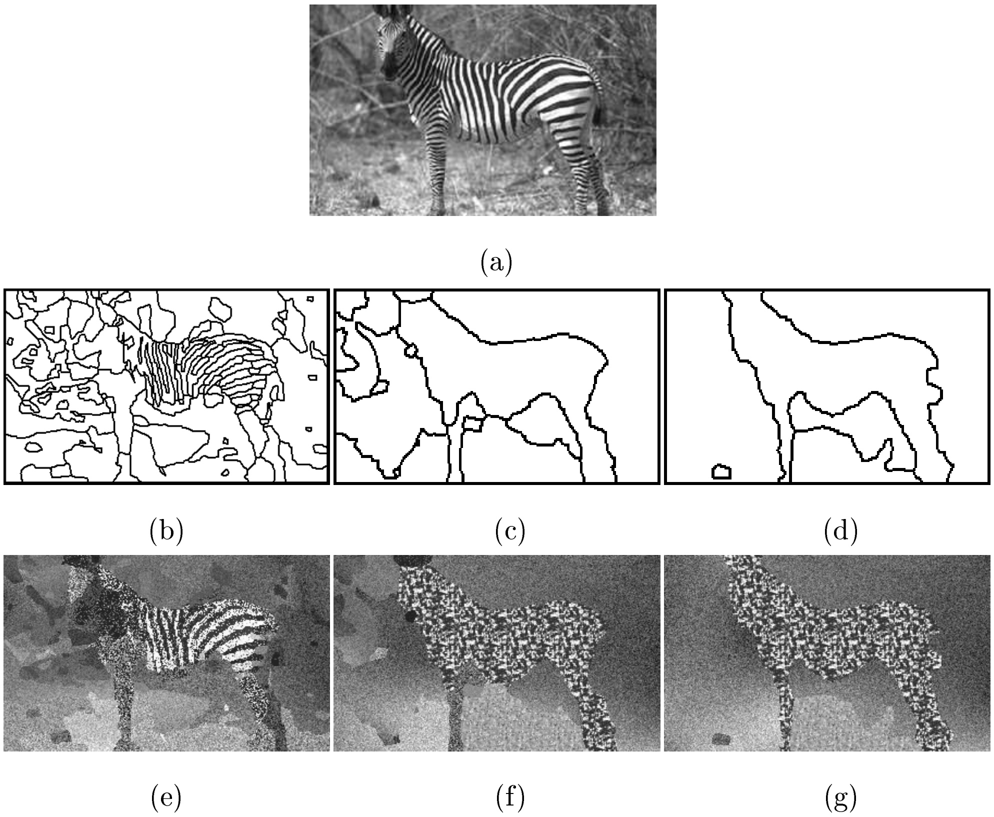


Figure 14 Experiments on the grey level zebra image with three solutions. (a) input image. (b)-(d) are three solutions, $W_i, i = 1, 2, 3$, for the zebra image. (e)-(g) are synthesized images $\mathbf{I}_i^{\rm syn} \sim p(\mathbf{I}|W_i^*)$ for verifying the results.

Figure 14 shows three segmentations on a grey level zebra image. As we discussed before, the DDMCMC algorithm in this paper has only one free parameter γ which is a "clutter factor" in the prior model (See equation (2)). It controls the extents of segmentations. A big γ encourages coarse segmentation with large regions. We normally extract results at three scales by setting $\gamma = 1.0, 2.0, 3.0$ respectively. In our experiments, the K-adventurers algorithm is effective only for computing distinct solutions in a certain scale. We expect the parameter γ can be fixed to a constant if we form an image pyramid with multiple scales and conduct segmentation with K-adventurers algorithm at each scale, and then propagate and refine the results to the next finer scale sequentially. This will be done in future research.

For the zebra image, W_1 segments out the black and white stripes while W_2 and W_3 treat the zebra as a texture region. The synthesized images $\mathbf{I}_i^{\text{syn}} \sim p(\mathbf{I}|W_i), i = 1, 2, 3$ show that the texture model is not sufficient because we choose only 8 small filters for computational ease. Also the spline surface model plays an important role in segmenting the ground and background grass, and this is verified by the global shading changes in $\mathbf{I}_2^{\text{syn}}$ and $\mathbf{I}_3^{\text{syn}}$.

Figures 15 and 16 display some other grey level and color images using the same algorithm. We show the input (left) and a segmentation (middle) starting with arbitrary initial conditions and a synthesized image (right) drawn from the likelihood $\mathbf{I}^{\text{syn}} \sim p(\mathbf{I}|W)$. The γ values for these images are mostly set up as 1.5 with a few obtained at 1.0-3.5. It took about 10-30 minutes, depending upon the complexity of image contents, on a Pentium III PC to segment an image with medium size such as 350×250 pixels after learning the pseudo-likelihood texture models at the beginning.

The synthesis images show that we need to engage more stochastic models such as point, curve process, and object like faces etc. For example, in the first row of Figure 16. The music band in a football stadium forms a point process which is not captured. The face is also missing in the synthesis.

Figure 17 shows three grey level images out of the 50 natural images in both color and grey level for the benchmark study. The input (left), the segmentation results by DDMCMC (middle), and the manual segmentation by a human subject (right) are displayed.

9 Discussion

In future work, we shall extend the DDMCMC paradigm in three directions.

1. Integrating other image models, such as point, curve processes for perceptual organi-

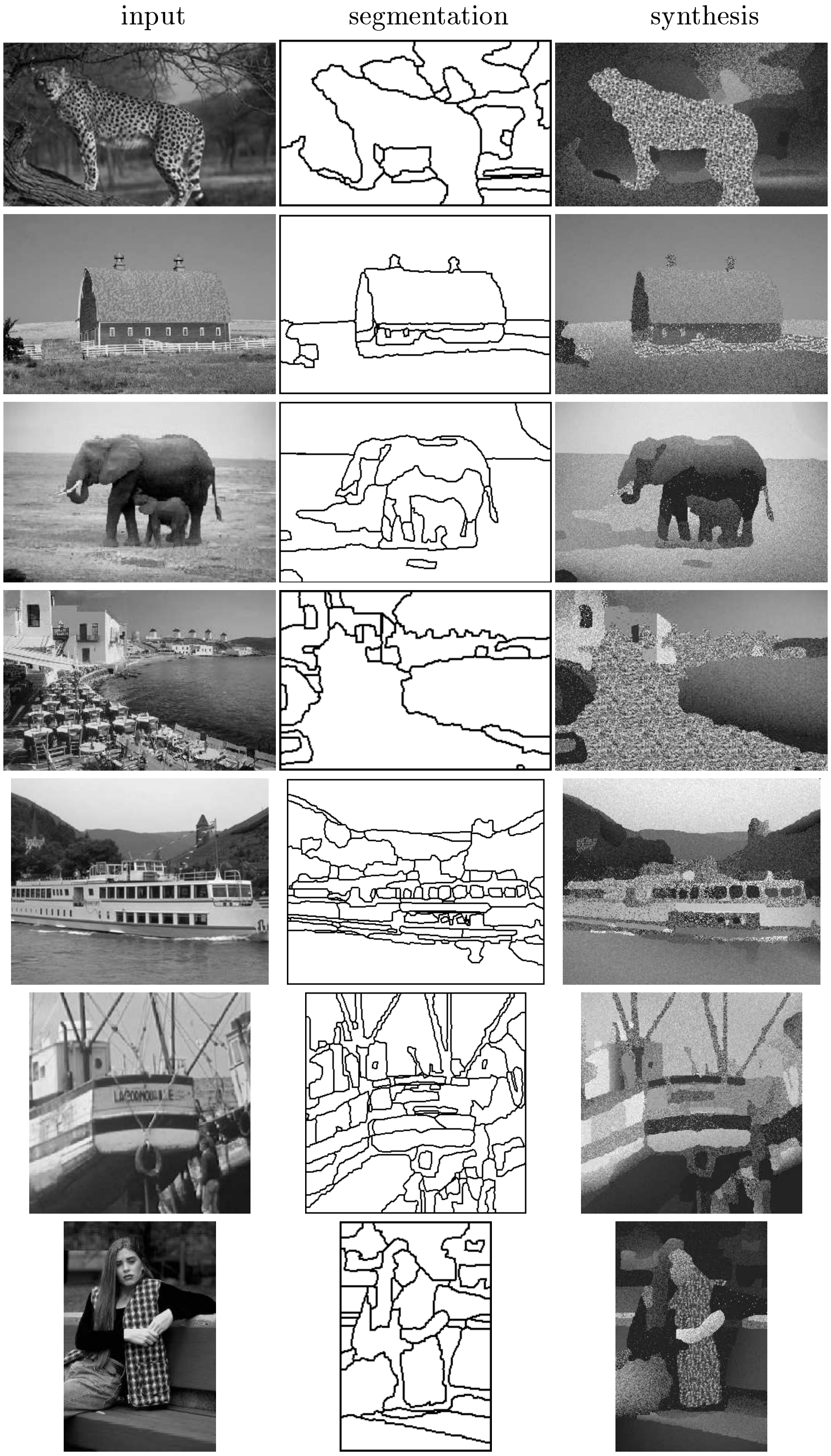


Figure 15 Grey level image segmentation by DDMCMC. Left: input images, middle: segmentation results W, right: synthesized images $\mathbf{I}^{\mathrm{syn}} \sim p(\mathbf{I}|W)$ with the segmentation results W.

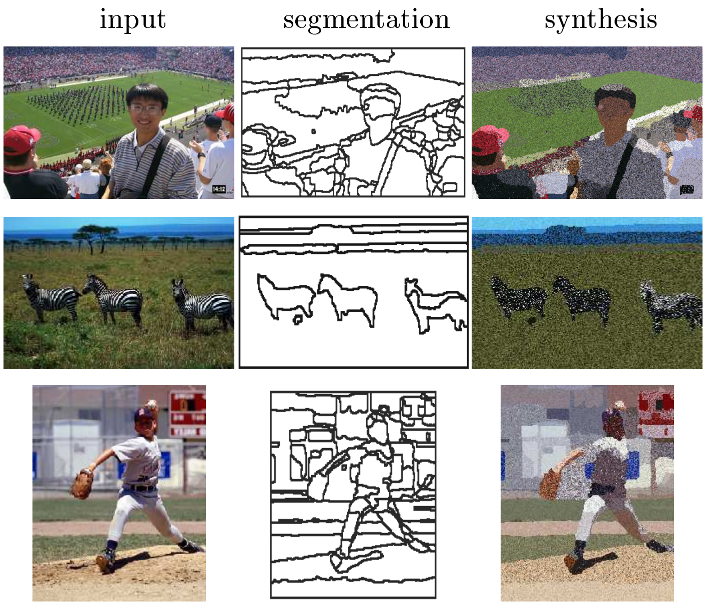


Figure 16 Color image segmentation by DDMCMC. Left: input images, middle: segmentation results W, right: synthesized images $\mathbf{I}^{\mathrm{syn}} \sim p(\mathbf{I}|W)$ with the segmentation results W.

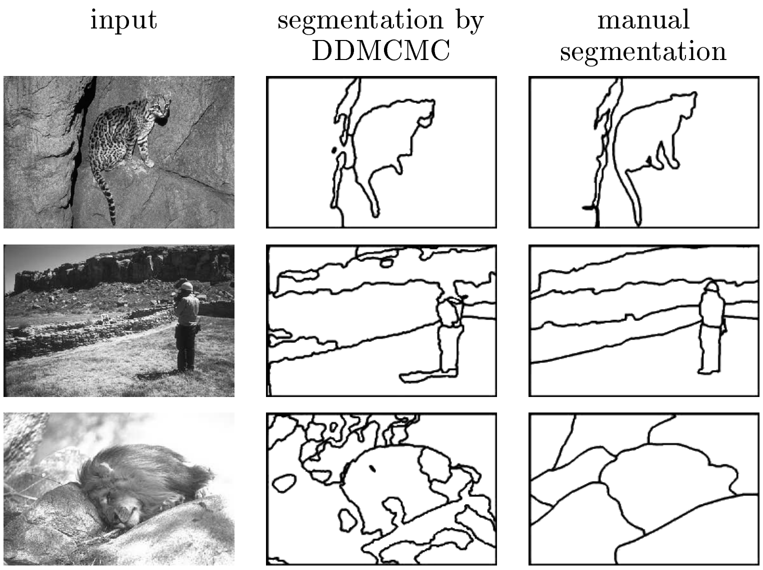


Figure 17 Some segmentation results by DDMCMC for the benchmark test by Martin. The errors for the above results by DDMCMC (middle) compared with the results by a human subject (right) are 0.1083, 0.3082, and 0.5290 respectively according to their metrics.

---

zation, and object models such as face, for object recognition.

2. Incorporating specific vision tasks with this stochastic inference engine. When there is a special purpose, the computing process is tuned (attended in a psychology term) to minimize some criterion, such as a risk function, which guides the selection and pruning of results.

3. Analyzing the DDMCMC convergence rate and linking it to the goodness of the set of heuristics qs.

Acknowledgments

This work is supported partially by two NSF grants IIS 98-77-127 and IIS-00-92-664, a NASA grant NAG-13-00039, an ONR grant N000140-110-535, and a Microsoft gift. A MS student Rong Zhang helps in the experiments for computing multiple solution experiments in 1D and 2D used in section (7). We thank David Mumford for discussions.

References

[1] L. Alvarez, Y. Gousseau, and J. M. Morel, "The Size of Objects in Natural Images," *Preprint of Centre de Math. and Applications*, ENS Cachan, Fr. 2000.

[2] S. A. Barker, A.C. Kokaram, and P.J.W. Rayner, "Unsupervised Segmentation of Images," *SPIE's 43rd Annual Meeting*, pp. 19-24, July 1998.

[3] C. Bouman and B. Liu, "Multiple Resolution Segmentation of Textured Images," *IEEE Trans. PAMI*, vol. 13, no. 2, pp. 99-113, Feb. 1991.

[4] J. F. Canny, "A Computational Approach to Edge Detection," *IEEE Trans. PAMI*, vol. 8, no. 6, pp. 679-698, Nov. 1986.

[5] Y. Cheng, "Mean Shift, Mode Seeking, and Clustering," *IEEE Trans. PAMI*, vol. 17, no. 8, pp. 790-799, Aug. 1995.

[6] D. Comaniciu and P. Meer, "Mean shift analysis and applications," *Proc. Int'l Conf. Comp. Vis.*, pp. 1,197-1,203, 1999.

[7] A. P. Dempster, N.M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *J. of the Royal Stat. Soc. series B*, 39:1-38, 1977.

[8] Y. N. Deng, B. S. Manjunath, and H. Shin, "Color Image Segmentation," *Proc. IEEE Conf. Comp. Vis. and Patt. Recog.*, pp. 446-451, 1999.

[9] D. A. Forsyth, "Sampling, Resampling and Colour Constancy," *Proc. IEEE Conf. Comp. Vis. and Patt. Recog.* pp. 300-305, 1999.

[10] S. Geman and D. Geman, "Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images," *IEEE Trans. PAMI*, vol. 6, pp. 721-741, Nov. 1984.

[11] P. J. Green, "Reversible Jump Markov Chain Monte Carlo Computation and Bayesian Model Determination," *Biometrika*, vol. 82, no. 4, pp. 711-732, 1995.

[12] U. Grenander and M. I. Millter, "Representations of Knowledge in Complex Systems," *J. of the Royal Stat. Soc. Series B*, vol. 56, no. 4, pp.549-603, 1994.

[13] M. Kass, A. Witkin and D. Terzopoulos, "Snakes: Active Contour Models," *Int'l J. Computer Vision*, vol. 1, no. 4, pp. 321-332, Jan. 1988.

[14] G. Koepfler, C. Lopez and J. M. Morel, "A Multiscale Algorithm for Image Segmentation by Variational Approach," *SIAM J. Numerical Analysis*, 31(1), pp. 282-299, 1994.

[15] Y. G. Leclerc, "Constructing Simple Stable Descriptions for Image Partitioning," *Int'l J. Comp. Vis.*, vol. 3, no. 1, pp. 73-102, 1989.

[16] A. B. Lee, D. B. Mumford, and J. G. Huang, "Occlusion Models for Natural Images," *Int'l J. Computer Vision*, vol. 41, pp. 35-59, Jan. 2001.

[17] H. C. Lee and D. R. Cok, "Detecting Boundaries in A Vector Field," *IEEE Trans. Signal Processing*, vol. 39, no. 5, pp. 1,181-1,194, 1991.

[18] K.L. Mengersen and R.L. Tweedie, "Rates of convergence of the Hastings and Metropolis algorithms," *Annals of Statistics*, 24:101-121, 1994.

[19] N. Metropolis, M.N. Rosenbluth, A.W. Rosenbluth, A.H. Teller, and E. Teller, "Equations of state calculations by fast computing machines", *J. Chem. Phys.* 21, 1087-92, 1953.

[20] D. B. Mumford and B. Gidas, "Stochastic models for generic images," *Quarterly of Applied Mathematics*, vol. LIX, no.1, pp. 85-111, March, 2001.

[21] D. Martin, C. Fowlkes, D. Tal, and J. Malik, "A Database of Human Segmented Natural Images and its Application to Evaluating Segmentation Algorithms and Measuring Ecological Statistics," *Proc. Int'l Conf. Computer Vision*, vol. 2, pp. 416-423, 2001.

[22] S. Oe, "Texture Segmentation Method by Using Two-Dimensional AR Model And Kullback Information," *Pattern Recognition*, vol. 26, pp. 237-243, 1993.

[23] S. Osher and J.A. Sethian, "Front propagating with curvature dependent speed: algorithms based on Hamilton-Jacobi formulation," *J. of Computational Physics*, vol.79, pp12-49, 1988.

[24] N. Paragios and R. Deriche, "Coupled Geodesic Active Regions for Image Segmentation: A Level Set Approach," *Proc. European Conf. Computer Vision*, Jun. 2000.

[25] S. Sclaroff and J. Isidoro, "Active Blobs," *Proc. Int'l Conf. Computer Vision*, pp. 1,146-1,153, 1998.

[26] J. Shi and J. Malik, "Normalized Cuts and Image Segmentation," *IEEE Trans. PAMI*, vol. 22, no. 8, Aug. 2000.

[27] R. H. Swendsen and J. S. Wang, "Nonuniversal Critical Dynamics in Monte Carlo Simulation," *Physical Review Letters*, vol. 58, no. 2, pp. 86-88, 1987.

[28] J. P. Wang, "Stochastic Relaxation on Partitions with Connected Components and Its Applications to Image Segmentation," *IEEE Trans. PAMI*, vol. 20, no. 6, pp. 619-636, June 1998.

[29] S. C. Zhu and A. L. Yuille, "Region Competition: Unifying Snakes, Region Growing, and Bayes/MDL for Multi-band Image Segmentation," *IEEE Trans. PAMI*, vol. 18, no. 9, pp. 884-900, Sept. 1996.

[30] S. C. Zhu, Y. N. Wu, and D. Mumford, "Filters, Random Field and Maximum Entropy: Towards a Unified Theory for Texture Modeling," *Int'l J. Computer Vision*, vol. 27, no. 2, pp. 107-126, 1998.

[31] S. C. Zhu and X. Liu, "Learning in Gibbsian Fields: How Accurate and How Fast Can It Be?" *Proc. IEEE Conf. Comp. Vis. and Patt. Recog.*, vol. 2, pp. 2-9, 2000.

Appendix: Approximating the KL-divergence

For simplicity of notation, we denote by $p(\mathbf{x})$ an arbitrary distribution in space Ω. In our case $p(\mathbf{x})$ represents the posterior $p(W|\mathbf{I})$.

For segmentation problems, we observe that $p(\mathbf{x})$ has two important properties.

1. $p(\mathbf{x})$ has an enormous number of local maxima (called modes in statistics). A significant mode corresponds to a distinct interpretation of the image, and the cloud surrounding a mode is local small perturbation of the region boundaries or model parameters. These significant modes of $p(\mathbf{x})$, denoted by $\mathbf{x}_i, i = 1, 2, ...,$ are well separated from each other due to the high dimensions.

2. Each mode $\mathbf{x}_i$ has a weight $\omega_i = p(\mathbf{x}_i)$, and its energy is defined as $E(\mathbf{x}_i) = -\log p(\mathbf{x}_i)$. The energies of these modes are uniformly distributed in a broad range $[E_{\min}, E_{\max}]$, say, $[1000, 10, 000]$. For example, it is normal to have solutions (or local maxima) whose energies differ in the order of 500 or more. Thus their probability (weights) differ in the order of $\exp^{-500}$, and our perception is interested in those "trivial" local modes!

Intuitively, it helps to imagine that $p(\mathbf{x})$ in Ω is distributed like the mass of the universe. Each star is a mode as local maximum of the mass density. The significant and developed stars are well separated apart from each other and their masses could differ in many orders of magnitudes. The above metaphor leads us to a mixture of Gaussian representation of $p(\mathbf{x})$. For a large enough N, we have,

$$p(\mathbf{x}) = \frac{1}{\boldsymbol{\omega}} \sum_{j=1}^{N} \omega_j G(\mathbf{x} - \mathbf{x}_j, \sigma_j^2), \quad \boldsymbol{\omega} = \sum_{j=1}^{N} \omega_j.$$

We denote by

$$S_o = \{(\omega_j, \mathbf{x}_j),\ j = 1, 2, ..., N\}, \quad \sum_{j=1}^{N} \omega_j = \boldsymbol{\omega},$$

the set of weighted particles (or modes). Thus our task is to select $K << N$ particles S from S_o. We define a mapping from the index in S to the index in S_o,

$$\tau : \{1, 2, ..., K\} \longrightarrow \{1, 2, ..., N\}.$$

Therefore,

$$S = \{(\omega_{\tau(i)}, \mathbf{x}_{\tau(i)});\ \ i = 1, 2, ..., K\}$$

S encodes a non-parametric model for approximating $p(\mathbf{x})$ by

$$\hat{p}(\mathbf{x}) = \frac{1}{\boldsymbol{\alpha}} \sum_{i=1}^{K} \omega_{\tau(i)} G(\mathbf{x} - \mathbf{x}_{\tau(i)}, \sigma^2_{\tau(i)}), \quad \boldsymbol{\alpha} = \sum_{i=1}^{K} \omega_{\tau(i)}.$$

Our goal is to compute

$$S^* = \arg \min_{|S|=K} D(p||\hat{p}).$$

For notational simplicity, we assume all Gaussians have the same variance in approximating $p(\mathbf{x})$, $\sigma_j = \sigma, j = 1, 2, ..., N$. By analogy, all "stars" have the same volume, but differ in weight. With the two properties of $p(\mathbf{x})$, we can approximately compute $D(p||\hat{p})$ in the following. We start with an observation for the KL-divergence for Gaussian distributions.

Let $p_1(x) = G(\mathbf{x} - \mu_1; \sigma^2)$ and $p_2(x) = G(\mathbf{x} - \mu_2; \sigma^2)$ be two Gaussian distributions, then it is easy ti check that

$$D(p_1||p_2) = \frac{(\mu_1 - \mu_2)^2}{2\sigma^2}.$$

We partition the solution space Ω into disjoint domains

$$\Omega = \cup_{i=1}^{N} D_i, \quad D_i \cap D_j = \emptyset, \ \forall i \neq j.$$

D_i is the domain where $p(\mathbf{x})$ is decided by a particle $(\omega_i, \mathbf{x}_i)$. The reason for this partition is that the particles in S are far apart from each other in high dimensional space and their energy varies significantly as the two properties state. Within each domain D_i, it is reasonable to assume that $p(\mathbf{x})$ is dominated by one term in the mixture and the other $N-1$ terms are neglectable.

$$p(\mathbf{x}) \approx \frac{\omega_i}{\boldsymbol{\omega}} G(\mathbf{x} - \mathbf{x}_i; \sigma^2), \quad \mathbf{x} \in D_i, \ i = 1, 2, ..., N.$$

The size of D_i is much larger than σ^2. After removing $N-K$ particles in the space, a domain D_i is dominated by a nearby particle that is selected in S.

We define a second mapping function

$$c : \{1, 2, ..., N\} \rightarrow \{1, 2, ..., K\},$$

so that $\hat{p}(\mathbf{x})$ in D_i is dominated by a particle $\mathbf{x}_{\tau(c(i))} \in S_K$,

$$\hat{p}(\mathbf{x}) \approx \frac{\omega_{c(i)}}{\boldsymbol{\alpha}} G(\mathbf{x} - \mathbf{x}_{\tau(c(i))}; \sigma^2), \quad \mathbf{x} \in D_i, i = 1, 2, ..., N.$$

Intuitively, the N domains are partitioned into K groups, each of which is dominated by one particle in S_K. Thus we can approximate $D(p||\hat{p})$.

$$
\begin{aligned}
D(p||\hat{p}) &= \sum_{n=1}^{N} \int_{D_n} p(\mathbf{x}) \log \frac{p(\mathbf{x})}{\hat{p}(\mathbf{x})} d\mathbf{x} \\
&= \sum_{n=1}^{N} \int_{D_n} \frac{1}{\boldsymbol{\omega}} \sum_{i=1}^{N} \omega_i G(\mathbf{x}-\mathbf{x}_i;\sigma^2) \log \frac{\frac{1}{\boldsymbol{\omega}} \sum_{i=1}^{N} \omega_i G(\mathbf{x}-\mathbf{x}_i;\sigma^2)}{\frac{1}{\boldsymbol{\alpha}} \sum_{j=1}^{k} \omega_{\tau(j)} G(\mathbf{x}-\boldsymbol{\mu}_{\tau(j)};\sigma^2)} d\mathbf{x} \\
&\approx \sum_{n=1}^{N} \int_{D_n} \frac{\omega_n}{\boldsymbol{\omega}} G(\mathbf{x}-\mathbf{x}_n;\sigma^2) [\log \frac{\boldsymbol{\alpha}}{\boldsymbol{\omega}} + \log \frac{\omega_n G(\mathbf{x}-\mathbf{x}_n;\sigma^2)}{\omega_{\tau(c(n))} G(\mathbf{x}-\mathbf{x}_{\tau(c(n))};\sigma^2)}] d\mathbf{x} \\
&= \sum_{n=1}^{N} \frac{\omega_n}{\boldsymbol{\omega}} [\log \frac{\boldsymbol{\alpha}}{\boldsymbol{\omega}} + \log \frac{\omega_n}{\omega_{\tau(c(n))}} + \frac{(\mathbf{x}_n - \mathbf{x}_{\tau(c(n))})^2}{2\sigma^2}] \qquad (28) \\
&= \log \frac{\boldsymbol{\alpha}}{\boldsymbol{\omega}} + \sum_{n=1}^{N} \frac{\omega_n}{\boldsymbol{\omega}} [(E(\mathbf{x}_{\tau(c(n))}) - E(\mathbf{x}_n)) + \frac{(\mathbf{x}_n - \mathbf{x}_{\tau(c(n))})^2}{2\sigma^2}] = \hat{D}(p||\hat{p}).
\end{aligned}
$$

Equation (28) has some intuitive meanings. The second term suggests that each selected $\mathbf{x}_{\tau(c(i))}$ should have large weight $\omega_{\tau(c(i))}$. The third term is the attraction forces from particles in S_o to particles in S. Thus it helps to pull apart the particles in S_k and also plays the role of encouraging to choose particles with big weight like the second term. It can be shown that eqn (27) generalizes the traditional (MAP) estimator when $K = 1$.

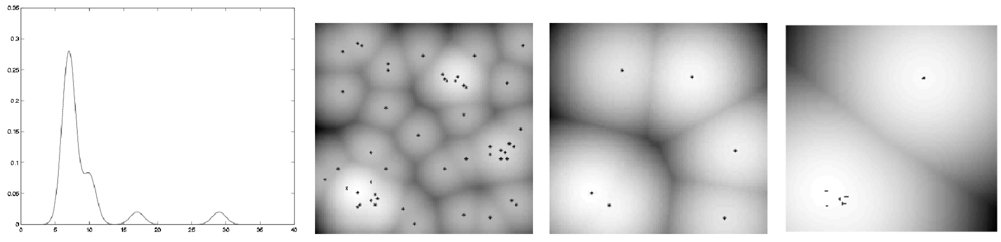
To demonstrate the goodness of approximation $\hat{D}(p||\hat{p})$ to $D(p||\hat{p})$, we show two experiments below.

Figure 18.a displays a 1D distribution $p(\mathbf{x})$ which is a mixture of $N = 4$ Gaussians (particles). We index the centers from left to right $\mathbf{x}_1 < \mathbf{x}_2 < \mathbf{x}_3 < \mathbf{x}_4$. Suppose we want to choose $K = 3$ particles for S_K and $\hat{p}(\mathbf{x})$. Table 1 displays the distances between $p(\mathbf{x})$ and $\hat{p}(\mathbf{x})$ over the four possible combinations. The second row shows the KL-divergence $D(p||\hat{p})$ and the third row is $\hat{D}(p||\hat{p})$. The approximation is very accurate given the particles are well separable.

Both measures choose (x_1, x_3, x_4) as the best S. Particle x_2 is not favored by the KL-divergence because it is near x_1, although it has much higher weight than x_3 and x_4. The fourth row shows the absolute value of the difference between $p(x)$ and $\hat{p}(x)$. This distance favors (x_1, x_2, x_3) and (x_1, x_2, x_4). In comparison, the KL-divergence favors particles that are apart from each other and picks up significant peaks in the tails.

chosen S_3:	$\{x_1, x_2, x_3\}$	$\{x_1, x_2, x_4\}$	$\{x_1, x_3, x_4\}$	$\{x_2, x_3, x_4\}$
$D(p\|\|\hat{p})$:	3.5487	1.1029	0.5373	2.9430
$\hat{D}(p\|\|\hat{p})$:	3.5487	1.1044	0.4263	2.8230
$\|p - \hat{p}\|$:	0.1000	0.1000	0.3500	1.2482

Table 1 Distances between $p(x)$ and $\hat{p}(x)$ for different particle set S_3.



(a) (b) (c) (d)

Figure 18 (a) A 1D dist. $p(\mathbf{x})$ with four particles $\mathbf{x}_i, i = 1, 2, 3, 4$. (b) A 2D dist $p(\mathbf{x})$ with 50 particles, we show $\log p(\mathbf{x})$ in the image for visualization. (c) $\hat{p}_1(x)$ with 6 particles that minimizes $D(p||\hat{p})$ or $D(p||\hat{p})$. (d) $\hat{p}_2(x)$ with 6 particles that minimizes $|p - \hat{p}|$.

This idea is better demonstrated in figure 18. Figure 18.a shows $\log p(\mathbf{x}) = -E(\mathbf{x})$ which is renormalized for displaying. $p(\mathbf{x})$ consists of $N = 50$ particles whose centers are shown by the black spots. The energies $E(\mathbf{x}_i), i = 1, 2, ..., N$ are uniformly distributed in an interval $[0, 100]$. Thus their weights differ in exponential order. Figure 18.b shows $\log \hat{p}(\mathbf{x})$ with $k = 6$ particles that minimize both $D(p||\hat{p})$ and $\hat{D}(p||\hat{p})$. Figure 18.c shows the 6 particles that minimize the absolute difference $|p - \hat{p}|$. Figure 18.b has more disperse particles.

For segmentation, a remaining question is: how do we measure the distance between two solutions W_1 and W_2? This distance measure is to some extent subjective. We adopt a distance measure which simply accumulate the differences for the number of regions in W_1, W_2, and the types ℓ of image models used at each pixel by W_1 and W_2.