Image Parsing by Data-Driven Markov Chain Monte Carlo

DISSEPTION

Presented in Partial Fulfillment of the Requirements for
the Degree Doctor of Philosophy in the
Graduate School of The Ohio State University

By

Zhuowen Tu, M.S.

* * * * *

The Ohio State University

2002

Dissertation Committee:

Professor Song-Chun Zhu, Adviser
Professor Han-Wei Shen, Co-Advisor
Professor Alan Y. Yuille

Approved by

Co-Adviser

Co-Adviser
Department of Computer and Information Science
ABSTRACT

It’s a scientific dream that we could build machines which understand the contents of natural images. The realization of this dream is challenged by the enormous complexity of natural images. The difficulty lies in two major aspects: (1) the modeling problem—how to model abundant patterns that emerge in images such as texture, lighting, shape, motion etc. (2) the computing problem—how to make inference of these patterns. Although the modeling problem has become increasingly clear by a number of recent developments [6, 49, 61, 65, 93, 96, 98], etc., designing a general and efficient computing framework remains a big challenge. Previous attempts in designing such a framework failed either because they are not general enough to deal with problems of high complexity or due to efficiency problem. We attack the computing problem by presenting a new paradigm called Data-Driven Markov Chain Monte Carlo (DDMCMC) in which exploration of the heterogeneous solution space is guided by importance proposals probabilistically in a efficient and robust manner. We make the contributions as follows: (1) We propose a stochastic computing paradigm called DDMCMC for visual inference and summarize many traditional vision problems for example, image segmentation, object recognition, perceptual organization etc., into a general problem called image parsing. It decomposes images into their natural patterns such as texture regions, curves, faces, etc. (2) In our first attempt to build an image parsing system, we use the DDMCMC paradigm and device a novel algorithm called image segmentation by DDMCMC wherein patterns of interest are regions of unknown
parameters from seven known types. This algorithm has been tested extensively for a large set of images. (3) We then introduce two generative curve models to extend the system to parse images into regions and curves. (4) Two high level patterns of structure, parallel curves and trees curves, are further incorporated into the system so that procedures of perceptual organization are carried out in an early stage of image parsing. (5) At the end of this thesis, we also provide some proofs of how to achieve efficient MCMC algorithms.
To my parents and wife
ACKNOWLEDGMENTS

I am grateful to my advisor Song-Chun Zhu whose most creative and original thinking have been my great source of inspiration. I feel specially indebted to him for sharing his thoughts with me, encouraging me for approaching a high level of scientific research and giving me financial support during my study in Computer Science department. His deep thinking and hard work made him one of the leading scientists in Computer Vision area. I’m also indebted to my former advisor Ron Li for bringing me into the field of Computer Vision and supporting me when I was pursuing a master degree at OSU.

My special thanks go to Prof. Han-Wei Shen, Prof. Alan Yuille, Prof. Kim Boyer and Prof. Raghu Machiraju for being in my Ph.D. dissertation committee and candidacy exam committee.

Without the unconditional and full support and love from lovely wife Pu Cui, my parents Shunxin Tu and Wenxian Zong, parents in law Yunsheng Wang and Sen Cui during the past years, it is hard to imagine that I could possibly begin, go through and finish my Ph.D. study. I owe them a great deal of both time and love.

I would also like to thank the members of the vision lab at Ohio State University, Cheng-en Guo, Feng Han, Adrian Barbu, Rong Zhang, Qiming Luo, Zijian Xu etc., the interaction with whom both intellectually and socially has been a highlight of my life.
VITA

October 17, 1972 .........................Born - P.R.China

1993 ........................................B.E. Beijing Information Technology Institute

1996 ............................................M.E. Tsinghua University

1999 ............................................M.S. The Ohio State University

1999-present .................................Graduate Research Associate, The Ohio State University.

PUBLICATIONS

Research Publications


Feng Han, Zhuowen Tu, and Song-Chun Zhu, A Stochastic Algorithm for 3D Scene Segmentation and Reconstruction. (submitted to) IEEE Trans. on Pattern Analysis and Machine Intelligence.


**FIELDS OF STUDY**

Major Field: Computer and Information Science
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Introduction</td>
<td>1</td>
</tr>
<tr>
<td>2.</td>
<td>Image Segmentation by Data-Driven Markov Chain Monte Carlo</td>
<td>7</td>
</tr>
<tr>
<td>2.1</td>
<td>Introduction</td>
<td>7</td>
</tr>
<tr>
<td>2.2</td>
<td>A Review of Existing Image Segmentation Algorithms</td>
<td>10</td>
</tr>
<tr>
<td>2.2.1</td>
<td>Some Traditional Methods</td>
<td>11</td>
</tr>
<tr>
<td>2.2.2</td>
<td>Normalized Cuts</td>
<td>12</td>
</tr>
<tr>
<td>2.2.3</td>
<td>Partitions with Connected Components</td>
<td>15</td>
</tr>
<tr>
<td>2.2.4</td>
<td>\textit{J-image} Segmentation</td>
<td>18</td>
</tr>
<tr>
<td>2.2.5</td>
<td>Image Segmentation by Reversible Jumps</td>
<td>19</td>
</tr>
<tr>
<td>2.3</td>
<td>Problem Formulation and Image Models</td>
<td>21</td>
</tr>
<tr>
<td>2.3.1</td>
<td>The Bayesian Formulation for Segmentation</td>
<td>22</td>
</tr>
<tr>
<td>2.3.2</td>
<td>The Prior Probability $p(W)$</td>
<td>23</td>
</tr>
<tr>
<td>2.3.3</td>
<td>The Likelihood $p(I</td>
<td>W)$ for Grey Level Images</td>
</tr>
<tr>
<td>2.3.4</td>
<td>Model calibration</td>
<td>26</td>
</tr>
</tbody>
</table>

Abstract ................................................................. ii

Dedication ............................................................... iv

Acknowledgments ....................................................... v

Vita ................................................................. vi

List of Tables ........................................................ xi

List of Figures ......................................................... xii

Chapters:
2.3.5 Image models for color ........................................ 28
2.4 Anatomy of Solution Space ..................................... 29
2.5 Exploring the Solution Space by Ergodic Markov chains .... 31
  2.5.1 Three basic criteria for Markov chain design. .......... 31
  2.5.2 Five Markov chain dynamics ............................... 32
  2.5.3 The bottlenecks ............................................. 34
2.6 Data-Driven Methods ............................................. 36
  2.6.1 Method I: clustering in atomic spaces $\varpi_\ell$ ....... 36
  2.6.2 Method II: Edge detection .................................. 41
2.7 Computing importance proposal probabilities ................. 43
  2.7.1 Computing importance proposal probability $q(\Theta | R, \ell)$ .... 44
  2.7.2 Computing importance proposal probability $q(\Gamma | R)$ ........ 45
  2.7.3 Computing $q(\Theta_i, \Theta_j | R, \ell_i, \ell_j)$ and $q(\Gamma_{ij} | R, \Theta_i, \Theta_j)$ .... 47
  2.7.4 A unifying framework ...................................... 48
2.8 Computing Multiple Distinct Solutions ......................... 49
  2.8.1 Motivation and a mathematical principle ................ 49
  2.8.2 A $K$-adventurers algorithm for multiple solutions .... 50
2.9 Experiments ......................................................... 52

3. Parsing Images into Regions and Additive Curves ............... 58
  3.1 Introduction ................................................... 58
  3.2 Previous curve models ...................................... 60
  3.3 Problem formulation in Bayes statistics .................... 61
  3.4 Probabilistic models for region processes .................. 63
  3.5 Probabilistic models for curve processes ................... 64
    3.5.1 A generative rope model of curve processes .......... 65
  3.6 The solution space $\Omega$ .................................... 67
  3.7 Integration of region and curve processes by MCMC ......... 69
    3.7.1 Dynamics design ....................................... 70
    3.7.2 Initialization by image pyramid and matching pursuit ... 72
    3.7.3 Summary of the algorithm .............................. 73
  3.8 Experiments .................................................... 73

4. Parsing Images into Regions and Occluded Curves ............... 79
  4.1 Introduction .................................................. 79
  4.2 Problem formulation in Bayes statistics .................... 80
  4.3 Probabilistic models for region and curve processes ....... 83
    4.3.1 Prior model for region processes .................... 83
    4.3.2 Prior model for curve processes ...................... 84
    4.3.3 Likelihood model for curve processes ................ 86
4.4 The solution space $\Omega$ .................................................. 87
4.5 Integration of region and curve processes by MCMC ................. 88
  4.5.1 Dynamics design ...................................................... 88
  4.5.2 Summary of the algorithm .......................................... 99
4.6 Experiments ................................................................. 100
4.7 Discussions ................................................................. 100

5. Grouping Curves into Parallel and Tree Structures by MCMC .......... 105
  5.1 Introduction ............................................................. 105
  5.2 Problem formulation in Bayes statistics ............................ 105
  5.3 Probabilistic models for region and curve processes ............... 108
  5.4 The solution space $\Omega$ .............................................. 110
  5.5 MCMC design .......................................................... 112
    5.5.1 A generalized graph labeling method ......................... 114
    5.5.2 Summary of the algorithm ..................................... 122
  5.6 Experiments ........................................................... 122

6. Analysis of DDMCMC Convergence Rate ................................. 127
  6.1 Bounds of convergence rate ......................................... 128
  6.2 Designing Transition Matrix $M$ With Big Conductance ............ 139

7. Future Work ............................................................... 147

Bibliography ................................................................. 148
LIST OF TABLES

Table ..........................  Page

3.1 A comparison of coding lengths in bytes for the example images by the proposed algorithm and the jpeg algorithm. .......................... 76
## LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>An example of image parsing and decomposition.</td>
<td>1</td>
</tr>
<tr>
<td>1.2</td>
<td>An example of two-layered image structure.</td>
<td>3</td>
</tr>
<tr>
<td>2.1</td>
<td>Four types of regions in the windows are typical in real world images.</td>
<td>24</td>
</tr>
<tr>
<td>2.2</td>
<td>Comparison study of four families of models. The first column is the original image regions cropped from four real world images shown in figure 2.1. The images in the 2-5 columns are synthesized images $r_{ij}^{syn} \sim p(I_{R}; \Theta_{ij}^*)$ 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.</td>
<td>27</td>
</tr>
<tr>
<td>2.3</td>
<td>The anatomy of the solution space. The arrows represent Markov chain jumps, and the reversible jumps between two subspace $\Omega_8$ and $\Omega_9$ realize a split-and-merge of a region. Each circle, triangle, square, and diamond represents the subspace for four types of models.</td>
<td>31</td>
</tr>
<tr>
<td>2.4</td>
<td>A color image and its clusters in $(L^<em>, u^</em>, v^*)$ space for $\alpha_{c1}$, the second row are six of the saliency maps associated with the color clusters.</td>
<td>38</td>
</tr>
<tr>
<td>2.5</td>
<td>(a)-(d) are saliency maps associated with four clusters in $\alpha_{c3}$. (e)-(h) are the color spline surfaces for the four clusters.</td>
<td>39</td>
</tr>
<tr>
<td>2.6</td>
<td>A clustering map (left) for $\alpha_{g1}$ and six saliency maps $S_i^{g1}$, $i = 1, \ldots, 6$ of a zebra image (input is in Fig. 2.14.a).</td>
<td>39</td>
</tr>
<tr>
<td>2.7</td>
<td>A clustering map (left) for $\alpha_{g2}$ and six saliency maps $S_i^{g2}$, $i = 1.6$ of a zebra image (input is in Fig. 2.14.a).</td>
<td>40</td>
</tr>
</tbody>
</table>
2.8 Texture clustering. A clustering map (left) and five saliency maps for five particles $\Theta^{*}_i$, $i = 1, 2, ..., 5$. ..................................................... 41

2.9 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. ..................................................... 42

2.10 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. ..................................................... 42

2.11 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. ..................................................... 43

2.12 A candidate region $R_k$ is superimposed on the partition maps at three scales for computing a candidate boundary $\Gamma_{ij}$ for the pending split. ........... 46

2.13 Segmenting a color image by DDMCMC with two solutions. See text for explanation. ..................................................... 52

2.14 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 $I^s^{syn} \sim p(I|W^*)$ for verifying the results. .... 53

2.15 Grey level image segmentation by DDMCMC. Left: input images, middle: segmentation results $W$, right: synthesized images $I^{syn} \sim p(I|W)$ with the segmentation results $W$. ................................................. 55

2.16 Color image segmentation by DDMCMC. Left: input images, middle: segmentation results $W$, right: synthesized images $I^{syn} \sim p(I|W)$ with the segmentation results $W$. ................................................. 56

2.17 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, 0.5290, and 0.7116 respectively according to their metrics. ......................... 57

3.1 An example of image parsing and decomposition. ....................... 58
3.2 Four existing curve models. a) SNAKE (Kass et al 1988), b) Random curves sampled from the Elastica model (Mumford, 1994), c) A random curve sampled from a MRF curve model (Zhu, 1999), d) A profile model for modeling faces (Cootes et al, 1995) and contour tracking (Isard and Blake, 1996).

3.3 A rope model of curves consists of a chain of knots. A knot has 1-3 image bases shown by the ellipses.

3.4 Base functions used for the disctionary $\Delta$.

3.5 Random ropes sampled from the prior model $p(C)$. a). The geometric curves for the sampled ropes. b). A symbolic representation of the sampled ropes. c). Curve layer $f^r$ by the ropes.

3.6 An input image $I$ (a) is initially decomposed into two components: $I^p_0$ (b) is a lowpass filtered version, and $I^s_0$ is the residue for high frequency components.

3.7 The algorithm that integrates the region and curve processes.

3.8 Result: parsing a sea shore image into regions and curves.

3.9 The segmentation and synthesis with region processes only for fig. 3.8.

3.10 More image parsing results by the algorithms with the integration of region and curve processes and by the previous DDMCMC algorithm with region processes only.

4.1 An example of two-layer image structure.

4.2 An example of curve domains and region domains.

4.3 A generative curve model.

4.4 Random curves sampled from the prior model $p(C)$. a). The geometric curve centerline (skeleton) for the sampled curves. b). Boundaries of the sampled curves. c). Curve image synthesized by the sampled curves.

4.5 An example for the death and birth of a curve.
4.6 “Curvelets” obtained by a matching pursuit method for a sea shore image. . . 90
4.7 An example of change of curve vector $W^c$ by splitting a curve into two. . . 94
4.8 An example of growing a curve knot and shrinking of a curve knot. . . . . 95
4.9 A solution vector. ................................................................. 97
4.10 Example: parsing images into regions and curves. ........................... 101
4.11 Example: parsing images into regions and curves. ........................... 103
4.12 Example: parsing images into regions and curves. ........................... 104
5.1 An example of two-layered image structure with independent curves, parallel structures, and trees in the curve layer. ......................... 106
5.2 Random samples of parallel and tree structures drawn from prior $p(W^p)$ and $p(W^t)$. ................................................................. 110
5.3 A generalized graph labeling methods. The top left figure shows a status $W'$ where different sets of sites are labeled with different values shown by different shadings. The neighboring sites with the same label is assigned a bond indicating how likely the pair should be bonded. We then sample these bonds $P_{on}/P_{off}$ in the top right graph to obtain an intermediate graph in which sites are separated into several connected components. From this graph, we randomly sample one of the connected component with a new label. The top right figure displays the new graph where edges are updated according to the new status $W'$. The same procedures can be applied if we want go from the status shown in the top right figure to the status shown in the top left figure. The bottom figure demonstrates the same stage into which both $W$ and $W'$ enter with one to one correspondence as the reason why the cancellation works. ....................................................... 116
5.4 An example of split-merge parallel curve group. ............................... 118
5.5 An example of splitting and merging tree curve groups. .................... 120
5.6 An example of splitting and merging tree curve groups. .................... 121
5.7 Result: parsing a sea shore image into regions and curves. .................... 123
5.8  Result: parsing a sea shore image into regions and curves. . . . . . . . . . . 124

5.9  Result: parsing a sea shore image into regions and curves. . . . . . . . . . . 125

5.10 Result: parsing a sea shore image into regions and curves. . . . . . . . . . . 126
### LIST OF ABBREVIATIONS

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCMC</td>
<td>Markov Chain Monte Carlo</td>
</tr>
<tr>
<td>DDMCMC</td>
<td>Data-Driven Markov Chain Monte Carlo</td>
</tr>
<tr>
<td>PDE</td>
<td>Partial Differential Equations</td>
</tr>
<tr>
<td>EM</td>
<td>Expectation Minimization</td>
</tr>
<tr>
<td>DOG</td>
<td>Difference of Gaussians</td>
</tr>
<tr>
<td>DOOG</td>
<td>Difference of Offset Gaussians</td>
</tr>
</tbody>
</table>
CHAPTER 1

Introduction

Figure 1.1: An example of image parsing and decomposition.

It’s a scientific dream that we could build machines which understand the contents of natural images. The realization of this dream is challenged by the enormous complexity of natural images that consist of a wide variety of stochastic patterns. Figure 1.1 demonstrates an example image which is decomposed into several layers of visual patterns such as
point, curve, color region, face, etc. The difficulty lies in two major aspects: (1) the **modelling** problem—how to model abundant patterns that emerge in images. (2) the **computing** problem—how to make inference of these patterns. Though the problem of modeling has become increasingly clear by a number of recent developments [6,49,61,65,93,96,98], etc. designing a general and efficient computing framework remains a big challenge. Previous attempts in designing such a framework failed either because they are not general enough to deal with problems of high complexity or due to efficiency problem and, thus, are not pragmatically feasible. Existing methods can be roughly divided into several categories: (1) Neural networks and Bayesian networks approaches like [42,70,71,74] are limited by the scopes of functions they can optimize and critically rely on the initial status from which the algorithms start. (2) Traditional pattern recognition techniques such as support vector machine [24], clustering [19], Neyman-Pearson Criterion [63] and many others are very hard to be applied in cases where solutions lie in complex space of multiple dimensions. (3) Partial Differential Equation (PDE) approaches [66,92] reach only local optimal status and require models to be known a prior. (4) Divide-and-Conquer and greedy algorithms [41,55,79,80,89] only work for certain functions and have difficulty in providing satisfactory results in general cases. (5) Traditional MCMC algorithms such as [35–37,83] suffer the speed problem and are not pragmatically feasible for visual inference.

We tackle the computing problem by presenting a new paradigm called Data-Driven Markov Chain Monte Carlo (DDMCMC) in which exploration of the solution space, an union of subspaces of varying dimensions, is guided by importance proposals probabilistically. The DDMCMC engine devised according to the paradigm plays a central role of integrating many bottom-up techniques such as, edge detection, clustering, and PDE,
which serve as either for proposing importance proposals or as jump and diffusion dynamics. Thus, it is well suited for solving a very difficult problem, image parsing, which decomposes images into their natural constituents and includes many traditional problems in vision, for example, object recognition, perceptual organization, and image segmentation. Our goal is to build a general image parsing system. Figure 1.2 illustrates a layered structure in which images are separated into two layers. The major focus of this dissertation is about to discover regions in the region layer (image segmentation), curves and curve groups in the curve layer. The outline of this thesis is:

1. In our first attempt to build an image parsing system, we use the DDMCMC paradigm to device a novel algorithm called image segmentation by DDMCMC wherein patterns of interest are regions of unknown parameters from seven known types. Four of them are used for modeling regions of the grey scale and three are used for modeling color regions. The four models account for regions of, namely, uniform, clutter, texture, and shading; The three models account for regions of, namely, uniform, texture,
and shading. This part is focused on the region layer as shown in figure 1.2 without considering the curve layer. The algorithm works as follows. Given an input image, clustering methods (mean-shift, EM) are first applied in the feature space for different model types to obtain clustering maps and the Canny edge detection algorithm is then used to detect edges boundaries at different scales followed by edge linking methods to create different segmentation maps. Segmentation is initialize with four rectangle regions with randomly selected image models. A central MCMC engine then randomly proposes to split or merge regions using obtained segmentation maps at different scale, switch region model using obtained clustering maps, diffuse region boundaries using the region competition algorithm [92]. The program stops when a certain criterion is met. We also design a method for selecting a set of results to preserve the intrinsic ambiguity of the image from the samples by MCMC. This algorithm has been tested extensively for a large set of images and is ranked the top by a recent benchmark study [57].

2. We then extend the DDMCMC image segmentation algorithm into a new system by introducing generative curve models with corresponding jump and diffusion dynamics such that it parses natural images into regions and curves. This part is focused on the region layer and the curve layer together with the assumption that all curves in the curve layer are independent without high level structures. To use the generative curve models, we test both an additive model and an occlusion model and find that the occlusion model works, in general, better than the additive model. For the additive model, we assume that a natural image amounts to the addition of the curve layer and the region layer. The curve layer consists of a number of curves each of which is
modeled as a chain of Difference of Gaussians (DOG) or Difference of Offset Gaussians (DOOG) bases. The region layer consists of an unknown number of regions that cover the entire image lattice, same as in the image segmentation. The task is to separate an input image into curve layer and region layer and discover curves and regions in each layer. The algorithm works as follows. Given an input image, clustering maps and segmentation maps are first obtained. The matching pursuit algorithm is applied to detect bases in an overcomplete basis as candidates. The curve layer is initialized with an empty set and the region layer is initialized with four rectangle regions. A central MCMC engine randomly proposes a move in either curve layer or region layer. In the curve layer, curves can be proposed to give birth, remove, split, and merge at random. In the region layer, the same procedures are applied as in the image segmentation. The program stops after a certain criterion is met.

For the occlusion model, we assume that a natural image amounts to the occlusion of curve layer and region layer with the curve layer sitting on top of the region layer. The curve layer consists of a number of curves each of which is modeled as a degenerated region with two parallel boundaries. Similarly, the task is to separate an input image into curve layer and region layer and discover curves and regions in each layer. The algorithm works in a similar manner as in the additive model except that curves and regions in the occlusion model have more interaction with each other.

3. Two high level patterns of structure, parallel curves and trees curves, are further incorporated into the system so that procedures of perceptual organization are carried out in an early stage of parsing. This part is focused on region layer and curve layer together with the assumption that there are independent curves, parallel curve groups and tree curve groups in the curve layer. The algorithm starts similarly as in the
previous part. What is different is that we design new MCMC dynamics to create, split, and merge parallel and tree curve groups.

4. We also provide some preliminary proofs of how to design issues of achieving efficient MCMC algorithms at the end of this thesis. The convergence speed of a Markov Chain is tied with the second largest absolute eigenvalue of its transition kernel.
CHAPTER 2

Image Segmentation by Data-Driven Markov Chain Monte Carlo

2.1 Introduction

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

The first challenge is the difficulty of modeling the 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 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 limited 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. It should output multiple distinct solutions dynamically so that these solutions “best
preserve” the intrinsic ambiguity. In our opinion, image segmentation should be considered a computing process not a vision task.

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 [46,48,92] with seven families of image models which compete to explain visual patterns in an image, for example, flat regions, clutter, texture, smooth shading, etc.

Secondly, we decompose the solution space into 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 2.3 for a space structure). The Bayesian posterior probability is distributed over this 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 [92], 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) [92].

Fourthly, we utilize data-driven techniques to guide the Markov chain search, and thus achieve tremendous speedups in comparison to previous MCMC algorithms [35–37]. In the literature, there are various techniques for improving the Markov chain speed, such as multi-resolution approaches [10,88], causal Markov models [10,64], and clustering
[5, 33, 39, 83, 88]. In our DDMCMC paradigm, data-driven methods such as edge detection [13] and tracing, data clustering [19, 26] are used. The results of these algorithms are expressed as weighted samples (or particles), which encode non-parametric probabilities in different 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 the “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 various hypotheses in the solution space.

Conceptually, the DDMCMC paradigm also give a new perspective on some well-known segmentation algorithms. Algorithms such as split-and-merge, region growing, Snake [45] and balloon/bubble [78], region competition [92], and variational methods [46], and PDEs [68] can be viewed as various MCMC jump-diffusion dynamics with minor modifications. Other algorithms, such as edge detection [13] and clustering [26, 29] can be used to 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. By comparing the synthesized images with the original images, we could easily judge, in some aspects, how good the segmentation results are and the sufficiency of image models. The ability of synthesizing images also allows to extend this segmentation algorithm into other applications such as image compression and image rendering.

2.2 A Review of Existing Image Segmentation Algorithms

Existing image segmentation algorithms can be divided into several categories:

1. Edge and contour based segmentation with or without edge linking by morphologic operator, perceptual organization criteria such as [13, 20, 21, 25, 45, 76], etc.

2. Region based segmentation by region growing [1, 73], split and merging procedures such as [8, 67].

3. Clustering methods such as [7, 26, 28, 43, 87]

4. The MDL (Minimum Description Length) approach, and energy minimization, level set method or Bayesian inferences, e.g. [10, 17, 30, 35, 44, 52, 64, 72].

5. Graph theory approaches such as Normalized-Cuts [54, 79, 80, 90]

Other techniques integrate several methods in different categories as in [68, 92]. Most of the above algorithms could be implemented in a multi-scale way, that is a coarse-to-fine approach [10, 52, 60]. In the literature, some state of art methods such as Canny edge operator [13], Mean-Shift clustering [26], and Region Competition [92] have been widely adopted in solving various vision problems. There also have been some recent developments on image segmentation systems that are aimed to deal with a large set of natural images. We shall give a brief review on these algorithms in the following sections.
2.2.1 Some Traditional Methods

Canny presented a landmark paper on how to design optimal operators for edge detection in [13]. The three criteria, namely, *Good detection*, *Good localization*, and *Only one response to a single edge*, have very intuitive interpretations. They are used to choose optimal edge operator for filtering edge signal embedded with noise. When applied, the first criteria accounts for suppression of noise w.r.t. edge signal at edge position, the second criteria discourages shift of edge position, and the third criteria prevents false edge positions to be detected. Canny used the first derivative of Gaussian to approximate the optimal operator and gave a method to detect edges in natural images. Apparently, by using edge linking and tracing methods such as [25], we can use Canny edge detector as an edge based segmentation algorithm, which will be used in chapter 2.

Mean-Shift [26], a clustering method, can be executed very efficiently for detecting local modes of the histogram of image feature, e.g. intensity values, for an image. An operator kernel with a fixed window size is randomly placed at a position in the feature space. For each step, the expectation of the kernel position is computed and is guaranteed to be closer to the local maximum than its center. Thus, the expectation is used as the new center of the kernel. By doing this, the kernel climbs up the histogram towards the local maximum and stops there. Once this mode has been detected, pixels whose value fall in the scope of the local model are taken off and their corresponding part in the histogram function is chopped off. Another local maximum can be detected by moving another randomly placed kernel function by repeating the above operations. The big challenge for the mean-shift algorithm is how to choose an appropriate window size for the kernel function and is usually unsolvable.
Region competition method [92], which will be used in chapter (2), integrates several types of segmentation algorithms such as region growing, MDL based algorithm, and Snake and Ballon methods. The algorithm is carried out by moving region boundary pixels in terms of two forces, specified as statistics and smoothness forces, which are obtained from the likelihoods and smoothness prior, respectively, by a Lagrangian approach. These two forces drive the boundary pixels to the places as to decrease the overall energy. This method heavily relies on having prior knowledge of region models and having good choices of region seeds and thus cannot be used to segment a large set of natural images. Other PDE based algorithms such as [66,68] are variational methods of region competition by evolving region boundaries using a level set approach. Although they overcome the problem of handling topological changes of regions naturally, region models are still need to be known a prior.

2.2.2 Normalized Cuts

The Normalized Cut framework was first introduced in [80] and the authors extended this framework with the incorporation of contour and texture to cope with general natural images in [54]. In [80], the authors first proposed a criterion, known as normalized cut, to divide a set of points into two disjoint groups if necessary. The image segmentation problem is treated as a graph partition problem. The original set is denoted as $V$, and is to be divided into two disjoint parts. $A, B, A \cup B = V, A \cap B = \emptyset$. The expense to make this cut is defined as the total weights between any two points in the different groups:

$$cut(A, B) = \sum_{u \in A, v \in B} w(u, v).$$

To achieve the minimal cut, one trivial solution will be one point in $A$ with the remaining points in $B$. The cut apparently needs to be normalized which is called the normalize
\[ Ncut(A, B) = \frac{cut(A, B)}{assoc(A, V)} + \frac{cut(A, B)}{assoc(B, V)} \]

where \( assoc(A, V) = \sum_{u \in A, t \in V} w(u, t) \) is the total connections from nodes in \( A \) to all nodes in the graph. They denote \( x \) as an \( N = |V| \) dimensional indicator vector, \( x_i = 1 \) if node \( i \) is in \( A \), and \(-1\) otherwise. Let \( d(i) = \sum_j w(i, j) \), be the total connection from node \( i \) to all other nodes. Let \( D \) be an \( N \times N \) diagonal matrix with \( d \) on its diagonal, \( W \) be an \( N \times N \) symmetrical matrix with \( W(i, j) = w_{ij}, k = \frac{\sum_{i \geq 0} d_i}{\sum_i d_i}, b = \frac{k}{1-k}, y = (1+x) - b(1-x) \). The \( Ncut \) can be rewritten as \( y^T(D-W)y \) whose minimal value is found when \( y \) is the second smallest eigenvector of matrix \( (D - W) \) (Rayleigh quotient). Thus, the proposed graph partition problem finds its answer in matrix theory and linear algebra.

The overall grouping algorithm was described as a Recursive 2-way \( Ncut \) method which consists of the following steps:

1. Given a set of features, set up a weighted graph \( G = (V, E) \), compute the weight on each edge, and summarize the information into \( W \), and \( D \).

2. Solve \((D - W)x = \lambda Dx\) for eigenvectors with the smallest eigenvalues.

3. Use the eigenvector with the second smallest eigenvalue to bipartition the graph by finding the splitting point such that \( Ncut \) is maximized,

4. Decide if the current partition should be sub-divided by checking the stability of the cut, and make sure that \( Ncut \) is below pre-specified value. Recursively repartition the segmented parts if necessary.
There are several important issues concerning this algorithm: computational time, transform of the second largest eigenvector from continuous values to an indicator vector, criterion to end the cut procedure, and the definition of the weight between every two points.

The authors found that the overall time complexity is just $O(n^{3/2})$ instead of $O(n^3)$ by observing that the weight matrix $W$ is a sparse matrix and sampling a number of neighboring points to compute weights of each point. The speed can be achieved about 2 minutes for a $100 \times 200$ image on a Pentium 200MHz computer. After the eigenvector corresponding to the second smallest eigenvalue is obtained, several thresholds to quantize vector with continuous values into binary indicator vector are picked and the $Ncut$ values of the corresponding indicator vector are computed among which the smallest is finally chosen. The weight of every two points is computed on both the spatial distance and feature distance of the two points using an exponential formula.

This *Normalized cuts* algorithm demonstrates some strengths in the problems of both image segmentation and perceptual organization. However, due to the discriminative model used for computing the weight matrix, each region doesn’t have a model to describe it. For some images, this algorithm, typically, will end up with over-segmentation on regions of smoothly changing intensities. It is also hard to incorporate Gestalt cues like parallelism and symmetry into the system. The partition procedure is not reversible and, thus, there is no way to merge two groups if segmented incorrectly, which is often the case. Many ad-hoc methods are used and hard thresholds are set in this algorithm and they lack convincing mathematical justifications. The eigen-based approach for partitioning a graph leaves many questions about whether it is an appropriate way for finding the global optimal solution. From the recently published results in Berkely’s benchmark comparison we can see that this algorithm gives many wrong segments.
In the spirit of the normalized cuts framework, the authors extended this work to incorporate contour and texture to deal with the segmentation of general natural images [54]. There are two basic families of image segmentation methods, contour based and region based approaches. While the contour based approaches may not have the knowledge of what is inside and has a big problem with textured regions, the region based approaches, by comparison, have a problem to locate the exact boundaries since the histograms of boundaries pixels may be similar to neither sides of the contour. The authors thus spend most of the efforts in analyzing how to combine the contour cue and texture cues together for the computation of the weight matrix. Although this is the first paper that is aimed at solving image segmentation of a large set of real images, it suffers the same problems as we discussed for the Ncut paper.

### 2.2.3 Partitions with Connected Components

Image segmentation by stochastic relaxation on partitions with connected components (PCC) [88] is another graph based approach. Instead of recursively bipartitioning the graph, it seeks to build a probability model on the graph and search for the optimal solution that maximizes the probability. Let \( L \) be the set of labels and \( S \) be the image lattice. The solution space of PCC, unlike a general labeling space \( \Omega = L^S \) which is huge and contains many unwanted configurations (like those in which disconnected regions are labeled with the same index), is made much more compact by labeling the components using four labels only. The idea was borrowed from the four-color problem: The connected regions of an arbitrary partition of a 2D lattice can be colored only by four colors such that two neighbors have different colors. The energy was then introduced as

\[
U(x) = \sum_i \sum_{s \in R_i} D(\xi(R_i), \xi(s)) - \gamma_1 |\partial_x x| + \gamma_2 |\delta x|
\]
where $x$ denotes a solution in the space $C(S)$ which is a subset of space of PCC, $\xi(s)$ is the image feature at site $s$, $\xi(R_i)$ is the mean of the features of every site in region $R_i$, $D(\xi(R_i), \xi(s))$ is a distance measure between $\xi(R_i)$ and $\xi(s)$, $|\partial_5x|$ accounts for the total number of pairs of sites on the different side of the region with feature distance larger than a threshold, $\delta$, $|\delta x|$ measures the total length of region boundaries. The overall goal is to minimize the energy $U(x)$. The first term in the above equation measures the homogeneity of each region so that all the pixels inside each region should have similar features, the second term encourages the appearance of boundary on the place where pixels have distinct features, and the third term favors compact regions with smooth boundaries. Instead of designing a diffusion dynamics for competing boundaries of all the regions, the author added the second term to encourage boundaries on pixels that have distinct feature values. Both Gibbs and Metropolis-Hasting algorithms were then discussed and were used. The dynamics need to be defined for sampling the probability by Gibbs or Metropolis-Hasting sampler. The basic requirements for defining such dynamics are reversibility and computability. That is, once one step is taken to change the $x$ to, let’s say $y$, there must possibly be another step for $y$ to come directly back to $x$, and the dynamics must be easy to compute and realize. The author designed only two dynamics that are even exclusively used as two different algorithms.

1. Randomly select one region $R$ and pick up one site $s$ in region $R$. If the removal of $s$ disconnects the remaining of the sites in $R$ then quit this move. Otherwise, combine the rest sites in $R$ as one region and either let $s$ become a region of its own or merge it with a randomly picked neighboring region.

2. Randomly pick up one region $R$ and pick up one site $s$ in region $R$. combine the remaining sites into regions (This time, there may be more than one because $s$ may
be the articulate point. An articulate point is defined as a point, the removal of which will disconnect the graph) and either let the s be a region of its own or merge s with the neighboring regions that are not neighbors.

The reversibility of the first scheme can easily be checked. If s doesn’t get merged with any of its neighbors, there will be a possibility that s will be picked up and merged with the one region created last time when s was taken out of R. In case s was merged with one of its neighbors, then s still got the chance to be picked up from that region in which s is impossible to be a articulate point so that this move won’t be forbidden and merged to the leftover of R last time. The attempt of the design of the second dynamic was to improve the efficiency of the sampler using the first dynamic at the expense of the complexity of the algorithm. The first dynamic, apparently, will leave the graph untouched and waste a trial if the s picked is, unfortunately, an articulate point.

The execution of the program is then carried out in a coarse-to-fine and fine-to-coarse fashion. The coarse resolution is first started from a large size of window for extracting features. The energy function is then defined at this scale. The program, however, is executed from fine-to-coarse, that is, pixels are combined into small connected components subject to minimizing the energy function and after a certain number of iterations these connected components are passed to the next stage where the basic elements to pick up are no longer pixels, instead, they are those connected components. The combined connected components can be further passed to the next the stage until a certain critria is met. Once the solution is reached at this scale, the boundaries pixels of each region are then freed, window size for feature extraction is decreased, and another energy function is defined separately. The process proceeds until the desired the scale is reached.
The merit of this algorithm is that it’s probability model is built on connected com-
ponents and reversibility is guaranteed at each stage. However, the overall reversibility is
not maintained once the connected components are formed at one stage and passed to the
next one. The energy functions are defined separately at each stage. Also, due to the idea
of using fixed basic elements (meta-regions) at each stage, boundary diffusion is not im-
plemented. Although the discriminative model used may be fast to compute, it lacks the
power of describing region thus will be limited to homogenous region segmentation only.
It can not handle large inhomogeneous region, such as sky, where intensities are smoothly
changing and no apparent edges exist.

2.2.4 J-image Segmentation

JSEG [29] algorithm segments a color image through two stages:

1. The color in the image is non-linearly quantized by a so called “peer group filtering
   method” and all the colors are then divided into several clusters (groups). Each clus-
ter is assigned with an unique id. A clustering map is obtained by labeling each pixel
   with its cluster id.

2. The second stage operates on the clustering map. For a region $R_i$, the centroid can
   be computed as $m_i = \frac{1}{N_i} \sum_{s \in R_i} s$ where $s$ is pixel position and $N_i$ is the number of
   pixels in region $R_i$. The variance is defined as $S_T = \sum_{s \in R_i} ||s - m_i||^2$. The average
   cluster variance is $S_W = \sum_{i=1}^{C} \sum_{s \in R_i, L(s)=i} ||s - m_i||^2$, where $C$ is the total number
   of clusters and $L(s)$ is the label of site $s$. The author introduced a term called $J$ with
   value $J = (S_T - S_W)/S_W$ which measures the homogeneity of a region. The idea is
   that a region with homogenous color should have colors evenly distributed thus the
   $S_T$ and the $S_W$ should be close, and they should be quite different otherwise. The
algorithm is aimed to minimize $\bar{J} = \frac{1}{N} \sum_k N_k J_k$ where $N_k$ is the number of pixels in region $k$. Although the authors say that “$\bar{J}$ was used as the criterion to be minimized over all possible ways of segmenting the image given the number of regions”, which was not justified, the number of regions is changing throughout the process. This is an apparent flaw since the minimal $\bar{J}$ can be reached by segmenting each single pixel as a region. Several $J$-images at different scales are computed by putting the center of different sizes of windows at each pixel to compute local $J$ for each pixel. At the coarse scale, some seeds are selected according the corresponding $J$-image and then grow to form compound regions. Neighboring regions are then merged according to the histograms on cluster labels. The similar process was carried out at finer scale.

This paper consists of several engineering hacks but works amazingly well and fast on many color images. There are several limitations of this paper. The quantization stage is not a well defined problem and should not be allowed to make hard decision since only the clustering map will be used afterwards in which color information is discarded. Thus, it could not handle regions with smoothly changing intensities. It also can not distinguish two regions with the same histogram of colors but distributed as different textures. The comparison of $\bar{J}$s is not valid since it is not well defined for different number of regions. The histogram distance measure used to merge regions is not accounted along with the change of $\bar{J}$ thus there lacks of a consistent criterion to work with to merge and put region seeds.

2.2.5 Image Segmentation by Reversible Jumps

Barker et al. used the partial decoupling method in sampling the posteriori probability to search the optimal solution for segmenting of images [5]. The author here work directly
with labeling each pixel with a region id of integer. The solution space for the region configuration thus is \( \Omega = \bigcup_{k=1}^{[\Lambda]} \{1, 2..k\}^\Lambda \), which is huge because not only the region number, \( k \), is unknown, but also each site \( s \) has \( k \) number of choices to be labeled. The region model, considered as texture, is described by Gaussian MRF (GMRF). The overall posterior probability is denoted as

\[
p(x, \Psi, k|y) \propto \exp\left\{ -\sum_{c \in \Lambda} \frac{1}{2} [y_c - \mu_c]^T \Sigma_c^{-1} [y_c - \mu_c] - \sum_{s \in \Omega} \beta V(x_s, x_{m_s}) \times p_r(\beta)p_r(k) \prod_{c \in \Lambda} p_r(\mu_c)p_r(\sigma_c)p_r(\Theta_c) \right\}
\]

where \( k \) is the number of region models forming the set \( \Lambda \), \( \Psi \) is the set of model parameters, and \( x \) is the labeling map. Using the Gibbs sampler to sample the probability mass in the solution space according to the above distribution is guaranteed to be a nightmare. The Swendsen-Wang algorithm is an improvement over the traditional Gibbs sampler by adding an auxiliary bond variables. The authors use the partial decoupling algorithm to further improve the speed, which unifies the Gibbs sampler and the Swendsen-Wang algorithm. The Gibbs sampler and the Swendsen-Wang algorithm sit on the two extremes of the partial decoupling algorithm. The algorithm works as follows:

1. Generate hyper-parameters for bond variable interaction in the partial decoupling process thus defining a priori the likely areas of 'homogeneity' for the clustering of region labels.

2. Randomly initialize the number of region models and the MRF state variables (or segmentation labels).

3. Set the initial annealing temperature: \( T = T_{MAX} \)
4. Sample the partial decoupling auxiliary bond variables given the MRF state configuration.

5. Sample the region label or MRF state configuration given the current auxiliary bond configuration.

6. Sample the region GMRF model parameters given the current region label realization.

7. Sample the number of region models $k$, i.e. the size of $\Lambda$ through the reversible jump sampling process.

8. Lower annealing temperature $T$ and if $T > T_{\text{MIN}}$

Although the authors tried the partial decoupling algorithm to speed up the sampling of the posterior probability, it seems still unrealistic to apply this algorithm to segment real images in which a large number of textured regions and inhomogeneous regions exist. Flipping pixels even with the assistance of bond variables is not an efficient way for the task of image segmentation which requires a group of many pixels to be played together and make decision on split or merge of regions efficiently. Without the explicit notion of regions, the prior for region boundaries is hard to be applied in this framework.

2.3 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.
2.3.1 The Bayesian Formulation for Segmentation

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

$$\Lambda = \bigcup_{i=1}^{K} R_i; \quad R_i \cap R_j = \emptyset, \quad \forall i \neq j. \quad (2.1)$$

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 [66,68] provides a good transform between the two.

Each image region $I_R$ is assumed to be coherent in the sense that $I_R$ is a realization from a probabilistic model $p(I_R; \Theta)$. $\Theta$ represents a stochastic process whose model type is indexed by $\ell$.

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

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

In a Bayesian framework, we make the inference about $W$ from $I$ over a solution space $\Omega$.

$$W \sim p(W|I) \propto p(I|W)p(W), \quad W \in \Omega.$$\(^1\) We transfer the (R,G,B) representation to $(L^*, u^*, v^*)$ for better color distance measure.
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.

### 2.3.2 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 $p(\Gamma) \propto e^{-\mu \int_{\partial R_i} ds}$.

3. A model for the size of each region. Recently both empirical and theoretical studies \cite{2, 49} 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

\[
p(A) \propto e^{-\gamma A^c},
\]

where $c = 0.9$ is a constant. $\gamma$ 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 \cite{62} 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 $\Theta$ of an image model penalizes model complexity in terms of the number of parameters $\Theta$, $p(\Theta|\ell) \propto e^{-\nu|\Theta|}$.

In summary, we have the following prior model

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

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

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 2.1 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.

![Figure 2.1: Four types of regions in the windows are typical in real world images.](image1)

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 $\omega_{g_1}, \omega_{g_2}, \omega_{g_3},$ and $\omega_{g_4}$ respectively. Let $G(0; \sigma^2)$ be a Gaussian density centered at 0 with variance $\sigma^2$.

\footnote{As a slight notation complication, $\Theta, \ell$ could be viewed as parameters or hidden variables in $W$. We use $p(I; \Theta, \ell)$ in both situations for simplicity.}
1. Grey image model family \( \ell = g_1 \): \( \omega_{g_1} \). This assumes that pixel intensities in a region \( R \) are subject to independently and identically distributed (iid) Gaussian distribution,

\[
p(I_R; \Theta, g_1) = \prod_{v \in R} G(I_v - \mu; \sigma^2), \quad \Theta = (\mu, \sigma) \in \omega_{g_1}.
\] (2.3)

2. Grey image model family \( \ell = g_2 \): \( \omega_{g_2} \). 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 \( j \).

\[
p(I_R; \Theta, g_2) = \prod_{v \in R} h(I_v) = \prod_{j=0}^{G} h_j^{n_j}, \quad \Theta = (h_0, h_1, ..., h_G) \in \omega_{g_2}.
\] (2.4)

3. Grey image model family \( \ell = g_3 \): \( \omega_{g_3} \). This is a texture model FRAME [93] with pixel interactions captured by a set of Gabor filters. This family of models was demonstrated to be sufficient for realizing a wide variety of texture patterns [94]. To facilitate the computation, we choose a set of 8 filters and formulate the model in pseudo-likelihood form [94]. The model is specified by a long vector \( \Theta = (\beta_1, \beta_2, ..., \beta_m) \in \omega_{g_3}, m \) is the total number of bins in the histograms of the 8 Gabor filtered images. Let \( \partial_v \) denote the Markov neighborhood of \( v \in R \), and \( h(I_v | I_{\partial v}) \) 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

\[
p(I_R; \Theta, g_3) = \prod_{v \in R} p(I_v | I_{\partial v}; \Theta) = \prod_{v \in R} \frac{1}{Z_v} \exp\{-<\Theta, h(I_v | I_{\partial v})>\},
\] (2.5)

\(<\cdot,\cdot>\) 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 \( \Theta \) can be estimated easily from images. We refer to a recent paper [94] for discussions on the computation of this model and its variants, such as patch likelihood, etc.
4. **Grey image model family** $g_4$: $\omega_{g_4}$. 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 $\Lambda$ (i.e. we fix the knots). Let $B(x,y)$ be the Bezier surface, for any $v = (x,y) \in \Lambda$,

$$B(x,y) = U^T_{(x)} \times M \times U_{(y)},$$

(2.6)

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

$$p(I_R; \Theta, g_4) = \prod_{v \in R} G(I_v - B_v; \sigma^2), \quad \Theta = (M, \sigma) \in \omega_{g_4}.$$

(2.7)

In summary, four types of models compete to explain a grey intensity region. Whichever fits the region better will have a higher likelihood. We denote by $\omega^g_\Theta$ the grey level model space,

$$\Theta \in \omega^g_\Theta = \omega_{g_1} \cup \omega_{g_2} \cup \omega_{g_3} \cup \omega_{g_4}.$$

### 2.3.4 Model calibration

The four image models should be calibrated to enable model selection 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 $\sim 100$ pixels, the data term dominates the prior and biases towards more complex models. Secondly, the pseudo-likelihood models in family $\omega_{g_2}$ are not a true likelihood
<table>
<thead>
<tr>
<th>observed</th>
<th>$\omega_{g_1}$</th>
<th>$\omega_{g_2}$</th>
<th>$\omega_{g_3}$</th>
<th>$\omega_{g_4}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mathbf{I}^\text{obs}_1$</td>
<td>$\mathbf{I}^\text{syn}_{11}$</td>
<td>$\mathbf{I}^\text{syn}_{12}$</td>
<td>$\mathbf{I}^\text{syn}_{13}$</td>
<td>$\mathbf{I}^\text{syn}_{14}$</td>
</tr>
<tr>
<td>$L_{11} = 1.957$</td>
<td>$L_{12} = 1.929$</td>
<td>$L_{13} = 1.689$</td>
<td>$L_{14} = 1.765$</td>
<td></td>
</tr>
<tr>
<td>$\mathbf{I}^\text{obs}_2$</td>
<td>$\mathbf{I}^\text{syn}_{21}$</td>
<td>$\mathbf{I}^\text{syn}_{22}$</td>
<td>$\mathbf{I}^\text{syn}_{23}$</td>
<td>$\mathbf{I}^\text{syn}_{24}$</td>
</tr>
<tr>
<td>$L_{21} = 3.563$</td>
<td>$L_{22} = 3.094$</td>
<td>$L_{23} = 2.749$</td>
<td>$L_{24} = 3.422$</td>
<td></td>
</tr>
<tr>
<td>$\mathbf{I}^\text{obs}_3$</td>
<td>$\mathbf{I}^\text{syn}_{31}$</td>
<td>$\mathbf{I}^\text{syn}_{32}$</td>
<td>$\mathbf{I}^\text{syn}_{33}$</td>
<td>$\mathbf{I}^\text{syn}_{34}$</td>
</tr>
<tr>
<td>$L_{31} = 3.852$</td>
<td>$L_{32} = 3.627$</td>
<td>$L_{33} = 2.514$</td>
<td>$L_{34} = 3.658$</td>
<td></td>
</tr>
<tr>
<td>$\mathbf{I}^\text{obs}_4$</td>
<td>$\mathbf{I}^\text{syn}_{41}$</td>
<td>$\mathbf{I}^\text{syn}_{42}$</td>
<td>$\mathbf{I}^\text{syn}_{43}$</td>
<td>$\mathbf{I}^\text{syn}_{44}$</td>
</tr>
<tr>
<td>$L_{41} = 3.121$</td>
<td>$L_{42} = 3.050$</td>
<td>$L_{43} = 1.259$</td>
<td>$L_{44} = 0.944$</td>
<td></td>
</tr>
</tbody>
</table>

Figure 2.2: Comparison study of four families of models. The first column is the original image regions cropped from four real world images shown in figure 2.1. The images in the 2-5 columns are synthesized images $\mathbf{I}^\text{syn}_{ij} \sim p(\mathbf{I}_R; \Theta^*_i)$ 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.

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 2.2 shows four typical images in the first column, which are cropped from
the images in Figure 2.1. We denote the four images by \( I_i^{\text{obs}}, i = 1, 2, 3, 4 \) on a lattice
\( \Lambda_0 \). For each image \( I_i^{\text{obs}} \), we compute its per pixel coding length (minus log-likelihood)
according to an optimal model within family \( \omega_{g_j} \) computed by a maximum likelihood
estimation for \( j = 1, 2, 3, 4 \).

\[
L_{ij} = \min_{\omega_{g_j} \in \Theta} \frac{\log p(I_i^{\text{obs}}; \Theta, g_j)}{|\Lambda_0|}, \quad \text{for } 1 \leq i, j \leq 4. \tag{2.8}
\]

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

\[
I_{ij}^{\text{syn}} \sim p(I_i; \Theta_{ij}^*, g_j), \quad \text{for } 1 \leq i, j \leq 4.
\]

We show \( I_i^{\text{obs}}, I_{ij}^{\text{syn}}, \) and \( L_{ij} \) in Figure 2.2 for \( 1 \leq i, j \leq 4 \).

The results in Figure 2.2 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 \( v \),

\[
\hat{p}(I_v; \Theta, g_j) = p(I_v; \Theta, g_j)e^{-c_j} \quad \text{for } j = 1, 2, 3, 4.
\]

\( c_j, j = 1, 2, 3, 4 \) 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 \( \omega_1, \omega_2, \omega_3, \) and \( \omega_4 \) respectively.

2.3.5 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** $c_1$: $\omega_{c_1}$. This is an iid Gaussian model in $(L^*, u^*, v^*)$ space.

\[
p(I_R; \Theta, c_1) = \prod_{\omega \in R} G(I_\omega - \mu; \Sigma), \quad \Theta = (\mu, \Sigma) \in \omega_{c_1}.
\]  

(2.9)

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

\[
p(I_R; \Theta, c_2) = \prod_{\omega \in R} [\alpha_1 G(I_\omega - \mu_1; \Sigma_1) + \alpha_2 G(I_\omega - \mu_2; \Sigma_2)].
\]

Thus $\Theta = (\alpha_1, \mu_1, \Sigma_1, \alpha_2, \mu_2, \Sigma_2) \in \omega_{c_2}$ are the parameters.

3. **Color image model family** $c_3$: $\omega_{c_3}$. We use three Bezier spline surfaces (see equation (2.6)) for $L^*$, $u^*$, and $v^*$ respectively to characterize regions with gradually changing colors such as sky, wall, etc. Let $B(x, y)$ be the color value in $(L^*, u^*, v^*)$ space for any $v = (x, y) \in \Lambda$,

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

Thus the model is

\[
p(I_R; \Theta, c_3) = \prod_{\omega \in R} G(I_\omega - B_\omega; \Sigma),
\]

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

In summary, three types of models compete to explain a color region. Whichever fits the region better will have higher likelihood. We denote by $\omega^c_\Theta$ the color model space, then

\[
\omega^c_\Theta = \omega_{c_1} \cup \omega_{c_2} \cup \omega_{c_3}.
\]

### 2.4 Anatomy of Solution Space

Before we design an algorithm, we need to study the structures of the solution space $\Omega$ on which the posterior probability $p(W|I)$ is distributed.
We start with the \textit{partition space} for all possible partitions of a lattice \( \Lambda \). When a lattice \( \Lambda \) is segmented into \( k \) disjoint regions, we call it a \textit{k-partition} denoted by \( \pi_k \),

\[
\pi_k = (R_1, R_2, \ldots, R_k), \quad \bigcup_{i=1}^{k} R_i = \Lambda, \quad R_i \cap R_j = \emptyset, \quad \forall i \neq j. \tag{2.10}
\]

If all pixels in each region are connected, then \( \pi_k \) is a connected component partition [88]. The set of all \( k \)-partitions, denoted by \( \varpi_{\pi_k} \), is a quotient space of the set of all possible \( k \)-colorings divided by a permutation group \( \mathcal{P} \mathcal{G} \) for the labels.

\[
\varpi_{\pi_k} = \{(R_1, R_2, \ldots, R_k) = \pi_k; \quad |R_i| > 0, \quad \forall i = 1, 2, \ldots, k\}/\mathcal{P} \mathcal{G}. \tag{2.11}
\]

Thus we have a general partition space \( \varpi_\pi \) with the number of regions \( 1 \leq k \leq |\Lambda| \),

\[
\varpi_\pi = \bigcup_{k=1}^{|\Lambda|} \varpi_{\pi_k}. 
\]

Then the solution space for \( W \) is a union of subspaces \( \Omega_k \), and each \( \Omega_k \) is a product of one \( k \)-partition space \( \varpi_{\pi_k} \) and \( k \) spaces for the image models

\[
\Omega = \bigcup_{k=1}^{|\Lambda|} \Omega_k = \bigcup_{k=1}^{|\Lambda|} \left[ \varpi_{\pi_k} \times \underbrace{\varpi_\Theta \times \cdots \times \varpi_\Theta}_{k} \right], \tag{2.12}
\]

where \( \varpi_\Theta = \bigcup_{i=1}^4 \varpi_{g_i} \) for grey level images, and \( \varpi_\Theta = \bigcup_{i=1}^3 \varpi_{c_i} \) for color images.

Figure 2.3 illustrates the structures of the solution space. In Figure 2.3, 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. \( \varpi_\Theta = \varpi_8^d \) is represented by a hexagon containing the four shapes. The partition space \( \varpi_{\pi_k} \) is represented by a rectangle. Each subspace \( \Omega_k \) 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 \( \Omega_k \) the \textbf{scene spaces}. \( \varpi_{\pi_k} \) 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 \( \Omega \) and thus are called the \textbf{atomic spaces}. Sometimes we call \( \varpi_\pi \) a \textbf{partition space} and \( \varpi_\ell, \ell = g_1, g_2, g_3, g_4, c_1, c_2, c_3 \) the \textbf{cue spaces}.  

30
Figure 2.3: The anatomy of the solution space. The arrows represent Markov chain jumps, and the reversible jumps between two subspace $\Omega_8$ and $\Omega_9$ realize a split-and-merge of a region. Each circle, triangle, square, and diamond represent the subspace for four types of models.

2.5 Exploring the Solution Space by Ergodic Markov chains

The solution space in Figure 2.3 is typical for vision problems. The posterior probability $p(W|I)$ 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.

2.5.1 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 [37]. 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 2.3.
Secondly, the Markov chain should be aperiodic. This is ensured by using the dynamics at random. The details can be seen in the next section.

Thirdly, the Markov chain has stationary probability \( p(W|I) \). This is replaced by a stronger condition of detailed balance equations which demands that every move should be reversible \([36, 37]\). The jumps in this paper all satisfy detailed balance and reversibility.

### 2.5.2 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 \([92]\). Let \( \Gamma_{ij} \) be the boundary between \( R_i, R_j, \forall i, j \), and \( \Theta_i, \Theta_j \) 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 \( \log p(W|I) \) plus a Brownian motion \( dB \) along the curve normal direction \( \bar{n}(s) \). By variational calculus, this is \([92]\),

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

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 small local minima. The log-likelihood ratio requires that the image models are comparable. Dynamics 1 realizes diffusion within the atomic (or partition) space \( \varpi_\pi \).
**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.

\[
\frac{d\Theta_i}{dt} = \frac{\partial \log p(I_{R_i}; \Theta_i, \ell_i)}{\partial \Theta_i}.
\]

This realizes diffusion in the atomic (or cue) spaces \( \omega, \ell \in \{ g_1, g_2, g_3, g_4, c_1, c_2, c_3 \} \) (move within a triangle, square, diamond, or circle of Figure 2.3).

**Dynamics 3-4: split and merge.** Suppose at a certain time step, a region \( R_k \) with model \( \Theta_k \) is split into two regions \( R_i \) and \( R_j \) with models \( \Theta_i, \Theta_j \), or vice versa, and this realizes a jump between two states \( W \) to \( W' \) as shown by the arrows in Figure 2.3.

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

where \( W_- \) denotes the remaining variables that are unchanged during the move. By the classic Metropolis-Hastings method [59], we need two proposal probabilities \( G(W \rightarrow dW') \) and \( G(W' \rightarrow dW) \). \( G(W \rightarrow dW') \) is a conditional probability for how likely the Markov chain proposes to move to \( W' \) at state \( W \), and \( G(W' \rightarrow dW) \) is the proposal probability for coming back. The proposed split is then accepted with probability

\[
\alpha(W \rightarrow dW') = \min(1, \frac{G(W' \rightarrow dW)p(W' | I)dW'}{G(W \rightarrow dW')p(W | I)dW}).
\]

There are two routes (or “pathways” in a psychology language) for computing the split proposal \( G(W \rightarrow dW') \).

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 \( \Gamma_{ij} \) 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 \( \ell_i \) and \( \ell_j \) 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 $\Theta_j$ with probability $q(\Theta_j | R_j, \ell_j)$. Thus,

$$ G(W \rightarrow dW') = q(3) q(R_k) q(\Gamma_{ij} | R_k) q(\ell_i) q(\Theta_i | R_i, \ell_i) q(\Theta_j | R_j, \ell_j) dW'. \quad (2.13) $$

In route 2, it first chooses two new region models $\Theta_i$ and $\Theta_j$, and then decides the boundary $\Gamma_{ij}$. Thus,

$$ G(W \rightarrow dW') = q(3) q(R_k) q(\ell_i) q(\ell_j) q(\Theta_i, \Theta_j | R_k, \ell_i, \ell_j) q(\Gamma_{ij} | R_k, \Theta_i, \Theta_j) dW'. \quad (2.14) $$

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,

$$ G(W' \rightarrow dW) = q(4) q(R_i, R_j) q(\ell_k) q(\Theta_k | R_k, \ell_k) dW. \quad (2.15) $$

$q(R_i, R_j)$ 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.

$$ W = (\ell_i, \Theta_i, W_{-}) \leftrightarrow (\ell_i', \Theta_i', W_{-}) = W'. $$

The proposal probabilities are

$$ G(W \rightarrow dW') = q(5) q(R_i) q(\ell_i') q(\Theta_i' | R_i, \ell_i') dW', \quad (2.16) $$

$$ G(W' \rightarrow dW) = q(5) q(R_i) q(\ell_i) q(\Theta_i | R_i, \ell_i) dW. \quad (2.17) $$

### 2.5.3 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 $q(1), \ldots, q(5)$,
\( q(R_k), q(R_4, R_5), q(\ell) \) 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.(2.13): Where is a good \( \Gamma \) for splitting a given region \( R \)? \( q(\Gamma|R) \) is a probability in the atomic (or partition) space \( \omega_\pi \).

2. \( q(\Theta|R, \ell) \) in eqns (2.13), (2.15) and (2.17): For a given region \( R \) and a model family \( \ell \in \{g_1, \ldots, g_4, c_1, c_2, c_3\} \), what is a good \( \Theta \)? \( q(\Theta|R, \ell) \) is a probability in the atomic (cue) space \( \omega_\ell \).

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 [92] and others [48]. 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 [58]

**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 \( p(W | I) \) within the atomic spaces \( \varpi_\pi \) and \( \varpi_\ell \) respectively. That is,

\[
q_\Gamma^* (\Gamma_{ij} | R_k) = p(\Gamma_{ij} | I, R_k), \quad q_\Theta^* (\Theta | R, \ell) = p(\Theta | I, R, \ell), \quad \forall \ell. \quad (2.18)
\]

Unfortunately, \( q_\Gamma^* \) and \( q_\Theta^* \) have to integrate information from the entire image \( I \), 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 \( \varpi_\ell \), \( \ell \in \{c_1, c_2, c_3\} \) and \( \ell \in \{g_1, g_2, g_3, g_4\} \) and edge detection in \( \varpi_\pi \). The results of clustering and edge detection are expressed as non-parametric probabilities for approximating the ideal marginal probabilities \( q_\Gamma^* \) and \( q_\Theta^* \) in these atomic space respectively.

### 2.6 Data-Driven Methods

#### 2.6.1 Method I: clustering in atomic spaces \( \varpi_\ell \)

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

\[
\mathcal{U}^\ell = \{ F^\ell_v : \ v \in \Lambda \}.
\]

In practice, \( v \) can be subsampled for computational ease. The set of vectors are clustered by either an EM method [28, 31] or a mean-shift clustering [19, 26] algorithm to \( \mathcal{U}^\ell \). The
EM-clustering approximates the points density in $\mathcal{U}^\ell$ 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 $\mathcal{U}^\ell$ and seeks the modes (local maxima) in its density (after some Gaussian window smoothing). Both algorithms return a list of $m$ weighted clusters $\Theta_1^\ell, \Theta_2^\ell, ..., \Theta_m^\ell$ with weights $\omega_i^\ell$, $i = 1, 2, ..., m$, and we denote by

$$\mathcal{P}^\ell = \{ (\omega_i^\ell, \Theta_i^\ell) : \ i = 1, 2, ..., m. \}.$$  \hfill (2.19)

We call $(\omega_i^\ell, \Theta_i^\ell)$ a weighted atomic (or cue) particle in $\mathcal{W}^\ell$ for $\ell \in \{c_1, c_3, g_1, g_2, g_3, g_4\}$.\footnote{The atomic space $\mathcal{W}_{c_2}$ is a composition of two $\mathcal{W}_{c_1}$, and thus is computed from $\mathcal{W}_{c_1}$.}

The size $m$ is chosen to be conservative, or it can be computed in a coarse-to-fine strategy with a limit $m = |\mathcal{W}^\ell|$. This is well discussed in the literature [19,26].

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

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

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

We call

$$S_i^\ell = \{ S_{i,v}^\ell : v \in \Lambda \}, \quad \text{for} \quad i = 1, 2, ..., m, \ \forall \ell$$  \hfill (2.20)

a saliency map associated with cue particle $\Theta_j^\ell$.

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

**Computing cue particles in $\mathcal{W}_{c_1}$**

For color images, we take $F_v = (L_v, U_v, V_v)$ and apply a mean-shift algorithm [19,26] to compute color clusters in $\mathcal{W}_{c_1}$. For example, Figure 2.4 shows a few color clusters (balls)
Figure 2.4: A color image and its clusters in \((L^*, u^*, v^*)\) space for \(\varpi_{c1}\), the second row are six of the saliency maps associated with the color clusters.

In a cubic \((L^*, u^*, v^*)\)-space for a simple color image (left), the size of the balls represents the weights \(\omega_i^{c1}\). Each cluster is associated with a saliency map \(S_i^{c1}\) 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.

**Computing cue particles in \(\varpi_{c3}\)**

Each point \(v\) contributes its color \(I_v = (L_v, U_v, V_v)\) as “surface heights”, and we apply an EM-clustering to find the spline surface models. Figure 2.5 shows the clustering result for the woman image. Figures 2.5.a-d are saliency maps \(S_i^{c3}\) for \(i = 1, 2, 3, 4\). Figures 2.5.e-h are the four reconstructed images according to fitted spline surfaces which recover some global illumination variations.
Figure 2.5: (a)-(d) are saliency maps associated with four clusters in $\varpi_{c_3}$. (e)-(h) are the color spline surfaces for the four clusters.

**Computing cue particles in $\varpi_{g_1}$**

In this model, the feature space $F_v = I_v$ is simply the intensity, and $\mathcal{U}^{i_1}$ 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 2.6 shows six saliency maps $S^{i_1}$, $i = 1, 2, ..., 6$ for a zebra image (the original image is shown in Figure 2.14.a. In the clustering map on the left in Figure 2.6, each pixel is assigned to its most likely particle.

Figure 2.6: A clustering map (left) for $\varpi_{g_1}$ and six saliency maps $S^{i_1}$, $i = 1, ..., 6$ of a zebra image (input is in Fig. 2.14.a).
Computing the cue particles in $\varpi_{g_2}$

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

Figure 2.7 shows the clustering results on the same zebra image.

![Clustering results](image)

Figure 2.7: A clustering map (left) for $\varpi_{g_2}$ and six saliency maps $S_i^{g_2}$, $i = 1..6$ of a zebra image (input is in Fig. 2.14.a).

Computing cue particles $\varpi_{g_3}$

At each subsampled pixel $v \in \Lambda$, we compute a set of 8 local histograms for 8 filters over a local window of $12 \times 12$ pixels. We choose 8 filters for computational convenience: one $\delta$ filter, two gradient filters, one Laplacian of Gaussian filter, and four Gabor filters. Each histogram has 9 bins. Then $F_v^{g_3} = (h_{v1,1}, \ldots, h_{v8,9})$ is the feature. An EM clustering is applied to find the $m$ mean histograms $\bar{f}_i$, $i = 1, 2, \ldots, m$. We can compute the cue particles for texture models $\Theta_i^{g_3}$ from $\bar{f}_i$ for $i = 1, 2, \ldots, m$. A detailed account of this transform is referred to a previous paper [94].
Figure 2.8: Texture clustering. A clustering map (left) and five saliency maps for five particles $\Theta_i^{y3}, i = 1, 2, ..., 5$.

Figure 2.8 shows the texture clustering results on the zebra image with one clustering map on the left, and five saliency maps for five particles $\Theta_i^{y3}, i = 1, 2, ..., 5$.

**Computing cue particles in $\varpi_{g4}$**

Each point $v$ contributes its intensity $I_v = F_v$ as a “surface height”, and we apply an EM-clustering to find the spline surface models. Figure 2.9 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.

**2.6.2 Method II: Edge detection**

We detect intensity edges using Canny edge detector [13] and color edges using a method in [50], and trace edges to form a partition of the image lattice. We choose edges at three 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 2.9: 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.

Figure 2.10.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 2.11 shows.

Figure 2.10: 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.
2.7 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 2.4-2.11 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 2.5.3. are computed from the data-driven results.
2.7.1 Computing importance proposal probability \( q(\Theta|R, \ell) \)

The clustering method in an atomic (cue) space \( \omega_\ell \) outputs a set of weighted cue particles \( P_\ell \). \( P_\ell \) encodes a non-parametric probability in \( \omega_\ell \),

\[
    q(\Theta|\Lambda, \ell) = \sum_{i=1}^{m} \omega_i^\ell G(\Theta - \Theta_i^\ell), \quad \text{with} \quad \sum_{i=1}^{m} \omega_i^\ell = 1, \tag{2.21}
\]

where \( G(x) \) is a Parzen window centered at 0. As a matter of fact, \( q(\Theta|\Lambda, \ell) = q(\Theta|I) \) is an approximation to a marginal probability of the posterior \( p(W|I) \) on cue space \( \omega_\ell \), \( \ell \in \{g_1, g_2, g_3, g_4, c_1, c_3\} \), since the partition \( \pi \) 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 \( S_i^\ell \) associated with \( \Theta_i^\ell \), i.e.,

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

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

\[
    \Theta \sim q(\Theta|R, \ell) = \sum_{i=1}^{m} p_i G(\Theta - \Theta_i^\ell). \tag{2.22}
\]

Equation (2.22) 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 \( \Theta_i^\ell \). Thus any \( \Theta \in \omega_\ell \) 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.
2.7.2 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” \( r^{(s)}_i, i = 1, 2, \ldots, n, \)

\[
\Delta^{(s)}(\Lambda) = \{ r^{(s)}_i : i = 1, 2, \ldots, n, \bigcup_{i=1}^{n} r^{(s)}_i = \Lambda \}, \quad \text{for } s = 1, 2, 3.
\]

These meta regions are then used in combination to form \( K \leq n \) regions \( R^{(s)}_1, R^{(s)}_2, \ldots, R^{(s)}_K, \)

\[
R^{(s)}_i = \bigcup_j r^{(s)}_j, \quad \text{with } r^{(s)}_j \in \Delta^{(s)}, \quad \forall i = 1, 2, \ldots, K.
\]

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

Let \( \pi^{(s)}_k = (R^{(s)}_1, R^{(s)}_2, \ldots, R^{(s)}_K) \) denote a \( k \)-partition based on \( \Delta^{(s)} \). \( \pi^{(s)}_k \) is different from the general \( k \)-partition \( \pi_k \) because regions \( R^{(s)}_i, i = 1, \ldots, K \) in \( \pi^{(s)}_k \) are limited to the meta-regions. We denote by \( \Pi^{(s)}_k \) the set of all \( k \)-partitions based on a partition map \( \Delta^{(s)} \).

\[
\Pi^{(s)}_k = \{ (R^{(s)}_1, R^{(s)}_2, \ldots, R^{(s)}_K) = \pi^{(s)}_k : \bigcup_{i=1}^{K} R^{(s)}_i = \Lambda \}. \quad \text{(2.23)}
\]

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

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

\[
q^{(s)}(\pi_k) = \frac{1}{|\Pi^{(s)}_k|} \sum_{j=1}^{|\Pi^{(s)}_k|} G(\pi_k - \pi^{(s)}_{k,j}), \quad \text{for } s = 1, 2, 3. \quad \forall k. \quad \text{(2.24)}
\]

\( G() \) is a smooth window centered at 0 and its smoothness accounts for boundary deformations and forms a cluster around each partition particle, and \( \pi_k - \pi^{(s)}_{k,j} \) measures the difference between two partition maps \( \pi_k \) and \( \pi^{(s)}_{k,j} \). Martin et al. [57] 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 \( \omega_{\pi_k} \). 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 \( \omega_{\pi_k} \),

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

This \( q(\pi_k) \) can be considered as an approximation to the marginal posterior probability \( p(\pi_k|\Omega) \).

![Figure 2.12: A candidate region \( R_k \) is superimposed on the partition maps at three scales for computing a candidate boundary \( \Gamma_{ij} \) for the pending split.](image)

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 2.12 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 \( \Gamma \).
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 2.12, $\Delta^{(1)}(R) = \{ r_1^{(1)}, r_2^{(1)} \}$, $\Delta^{(2)}(R) = \{ r_1^{(2)}, r_2^{(2)}, r_3^{(2)}, r_4^{(2)} \}$, and so on.

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

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

(2.25)

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

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

(2.26)

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 $c$.

### 2.7.3 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 (2.14)).
For example, there are two ways to perceive the zebra in Figure 2.14. One perceives
the zebra as one textured region (by a model in $\omega_{g_3}$). The other sees it as one region of
black stripes plus one region of white strips and thus uses two models in $\omega_{g_1}$ or $\omega_{g_4}$. The
Markov chain should be able to switch between the two perceptions effectively (see results
in Figure 2.14.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 $\Theta_i$ from the candidate set when we select $\Theta_j$. Then we decide on the
boundary $\Gamma q(\Gamma_{ij}| R, \Theta_i, \Theta_j)$ by randomly labeling the pixels in $R$ according to probabilities
of the saliency maps.

### 2.7.4 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 [13, 50] compute implicitly a marginal probability $q(\pi| I)$
on the partition space $\omega_{\pi}$. Secondly, clustering algorithms [19, 26] compute a marginal
probability on the model space $\omega_{\ell}$ for various models $\ell$. Thirdly, the split-and-merge and

2.8 Computing Multiple Distinct Solutions

2.8.1 Motivation and a mathematical principle

The DDMCMC paradigm samples solutions from the posterior \( W \sim p(W|I) \) 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 \textit{maximum a posteriori} (MAP) estimator

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

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 \( I \) 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 \textit{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 \( \Omega \). (Our result was early presented in a CVPR2000 paper).
Let \( S = \{ (\omega_i, W_i) : i = 1, \ldots, K \} \) be a set of \( K \) weighted solutions which we call “scene particles”, the weight is its posterior probability \( \omega_i = p(W_i|I), i = 1, 2, \ldots, K \). (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 \( \Omega \),

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

\( G \) is a Gaussian window in \( \Omega \).

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 \( \hat{p}(W|I) \) approaches \( p(W|I) \) by minimizing a Kullback-Leibler divergence \( D(p||\hat{p}) \) under a complexity constraint \(|S| = K\),

\[
S^* = \arg \min_{|S|=K} D(p||\hat{p}) = \arg \min_{|S|=K} \int p(W|I) \log \frac{p(W|I)}{\hat{p}(W|I)} dW. \tag{2.27}
\]

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

### 2.8.2 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 \( p(W|I) \) 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 \( p(W|I) \) 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. Similar idea of preserving multiple results could be found in [47] and receives more discussion in a discrepancy theory by Chazelle [16].
Here we present a greedy algorithm for computing $S^*$ approximately. We call the algorithm—“$K$-adventurers algorithm.\footnote{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.}

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 $S_{\text{new}}$ by minimizing the approximative KL divergence $\bar{D}(p_+ || p_{\text{new}})$.

**The k-adventurers algorithm**

1. Initializing $S$ and $\hat{p}$ by repeating one initial solution $K$ times.
2. Repeat
3. Compute a new particle $(\omega_{K+1}, x_{K+1})$ by DDMCMC after a successful jump.
4. $S_+ \leftarrow S \cup \{(\omega_{K+1}, x_{K+1})\}$.
5. $\hat{p} \leftarrow S_.$
6. For $i = 1, 2, \ldots, K + 1$ do
7. $S_{-i} \leftarrow S_+ / \{(\omega_i, x_i)\}$.
8. $\hat{p}_{-i} \leftarrow S_{-i}$.
9. $d_i = D(p || \hat{p}_{-i})$.
10. $i^* = \text{arg min}_{i \in \{1, 2, \ldots, K + 1\}} d_i$.
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.
2.9 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 [57] by the Berkeley group\(^6\), where the results by DDMCMC and other methods such as [80] 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.

![Diagram of DDMCMC process](image)

**Figure 2.13:** Segmenting a color image by DDMCMC with two solutions. See text for explanation.

We first show our working example on the color woman image. Following the importance proposal probabilities for the edges in Figure 2.10 and for color clustering in Figure 2.4, we simulated three Markov chains with three different initial segmentations shown

\(^5\)See [www.cis.ohio-state.edu/oval/Segmentation/DDMCMC/DDMCMC.htm](http://www.cis.ohio-state.edu/oval/Segmentation/DDMCMC/DDMCMC.htm)

in Figure 2.13 (top row). The energy changes ($-\log p(W|\mathbf{I})$) of the three MCMCs are plotted in Figure 2.13 against time steps. Figure 2.13 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 $\mathbf{I}_i^{\text{syn}} \sim p(\mathbf{I}|W_i), i = 1, 2$. The synthesis is a good way to examine the sufficiency of models in segmentation.

![Zebra image](image)

Figure 2.14: 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 $\mathbf{I}_i^{\text{syn}} \sim p(\mathbf{I}|W_i^{*})$ for verifying the results.

Figure 2.14 shows three segmentations on a grey level zebra image. As we discussed before, the DDMCMC algorithm in this paper has only one free parameter $\gamma$ which is a “clutter factor” in the prior model (See equation (2.2)). It controls the extents of segmentations. A big $\gamma$ 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 $\gamma$ 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 $\mathbf{I}_i^{\text{syn}} \sim p(I|W_i), i = 1, 2, 3$ 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 2.15 and 2.16 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 $\mathbf{I}^{\text{syn}} \sim p(I|W)$. The $\gamma$ 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 $350 \times 250$ 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 2.16. The music band in a football stadium forms a point process which is not captured. The face is also missing in the synthesis.

Figure 2.17 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.
Figure 2.15: Grey level image segmentation by DDMCMC. Left: input images, middle: segmentation results $W$, right: synthesized images $I^{\text{syn}} \sim p(I|W)$ with the segmentation results $W$. 
Figure 2.16: Color image segmentation by DDMCMC. Left: input images, middle: segmentation results $W$, right: synthesized images $I^\text{syn} \sim p(I|W)$ with the segmentation results $W$. 
Figure 2.17: 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, 0.5290, and 0.7116 respectively according to their metrics.
CHAPTER 3

Parsing Images into Regions and Additive Curves

3.1 Introduction

Figure 3.1: An example of image parsing and decomposition.

Natural images consist of a wide variety of stochastic visual patterns. As an example, Figure 3.1 illustrates how an image is decomposed into point, line, curve processes, regions of coherent color and textures, and objects. Parsing an image into its constituent

58
components is a fundamental problem in image understanding. It augments the current pixel-based image representation to a semantic object-based description, and thus has broad impacts on many applications, including image compression, photo editing, image database retrieval, object recognition in vision, and non-photo-realistic rendering (NPR) in graphics. Solving the image parsing problem needs 1) a number of types of probabilistic models which can characterize various visual patterns and are compatible (or comparable) with each other, and 2) inference algorithms which can effectively handle different types of probabilistic models and achieve globally optimal solution. These are general topics of this paper.

In the literature, image segmentation work usually assume that images contain regions of homogeneous properties, such as textures, textons (attributed points), colors, and shading. While many visual patterns are represented well by such 2D models, there are many other visual patterns which are fundamentally one dimensional and thus cause major problems in segmentation. For example, see those lines, trees, grids in Figures 3.6, 3.8, and 3.10. We call the 2D patterns region processes and 1D patterns curve processes.

In this thesis, we propose a stochastic algorithm for parsing an image into a number of region and curve processes. The paper makes contributions in the following aspects.

Firstly, it presents a generative rope model for curve patterns. Following the idea of a generative texture model proposed by Guo et al. 2001 [38], we propose our rope model here. The rope model is in the form of hidden Markov model. The hidden layer is a chain of connected knots. Each knot has 1-3 image bases selected from an over-complete basis [65], such as DOG (difference of Gaussians) or DOOG (difference of offset Gaussians) at various scales and orientations. The Markov model accounts for not only the geometric smoothness as in the SNAKE [45] or Elastica models [61], but also the photometric coherence along the curve.

Secondly, it integrates both 2D region and 1D curve models for image parsing. We adopt a generative image model which represents an image I as a superposition of two layers $I = I' + I^c$. $I'$ is partitioned into disjoint regions and $I^c$ is a linear sum of some image bases which are grouped into curves. The algorithm consists of two parts. Part I is
image segmentation on $I^c$, and this is done by a recent data driven Markov chain Monte Carlo (DDMCMC) algorithm [85]. Part II infers the curves patterns from $I^c$, and is the main focus of this paper. By virtue of the generative image models, the cooperation of the two parts is governed in the posterior probability.

Thirdly, it achieves global optimization by effective Markov chain Monte Carlo. Due to the use of different families of models, the solution space consists of many subspaces of varying dimensions. Therefore the Markov chain consists of reversible jumps [36, 37] and stochastic diffusions to explore the complex solution space. The jump dynamics realize the death/birth, split and merge of regions and curves, the switching of models, and so on, while the diffusion process realizes region growing/competition [92], and curve deformation [45].

The algorithm is applied to a set real images and some results are shown in Figures 3.8 and 3.10. We verify the results through random synthesis from the computed solution and compare the results against segmentations with region process only.

We organize this chapter as follows. We first formulate the problem in Section 3.3. The Section 3.4 briefly overviews the region models used by region processes. Section 3.5 discusses the rope model for curve patterns. Section 3.6 discusses the structure of the solution space. Then we present the integrated algorithm in Section 3.7. Some experiments are shown in section 3.8.

3.2 Previous curve models

There are four interesting curve models in the literature as Figure 3.2 shows. The first one is the SNAKE model [45]. Let $C(s), s \in [a, b]$ be a continuous curve, a SNAKE model has a smooth term plus an image gradient term for intensity discontinuity.

$$p(C) \sim \exp\{-\int_a^b [-|\nabla I|^2 + \alpha \tilde{C}(s) + \beta \hat{C}(s)] ds\}.$$
Mumford studied an Elastica model in [61] motivated by a Ulenbeck process for a moving particle with friction, let \( \kappa(s) \) be the curvature, then

\[
p(C) \sim \exp\left\{ - \int_a^b \beta + \alpha \kappa^2(s) \right\} ds.
\]

Figure 3.2.b shows some typical curves sampled from the above model. The third model was proposed by Zhu in [96] which integrates model Gestalt properties and symmetry in a Markov random field model for closed contours, and is learned by a maximal entropy principle. Figure 3.2.c shows a typical sample of curve from this model. The fourth model is used by Cootes et al. [22] for face contour and are also used in tracking [40]. It is a smooth contour but also measures image profiles along lines perpendicular to the contour as Figure 3.2.d shows. This profile is used to local edges nearby. Recently, August and Zucker [3] constituted a so called curve indicator random field model.

As we can see, the gradient term in the SNAKE model or the profile measure are both feature detectors, and they are not generative models for image intensities.

### 3.3 Problem formulation in Bayes statistics

Let \( \Lambda = \{(i, j) : 1 \leq i \leq L, 1 \leq j \leq H\} \) be an image lattice, and \( I \) be an intensity image defined on \( \Lambda \). We assume that the image \( I \) is a superposition of two layers,

\[
I = I^r + I^c, \tag{3.1}
\]

where \( I^r \) and \( I^c \) are called region layer and curve layer respectively.

We assume that the region layer \( I^r \) consists of a number of \( K^r \) disjoint regions which form a partition of the lattice \( \Lambda \).

\[
\bigcup_{i=1}^{K^r} R_i = \Lambda, \quad R_i \cap R_j = \emptyset, \quad \forall i \neq j.
\]
Let $I^r_R$ denote the image intensities in a region $R$, and a region is said to be coherent in the sense that $I^r_R$ is a realization from a probabilistic model $p_\ell(I^r_R; \Theta)$. $\ell$ indexes the model or a stochastic process, and $\Theta$ is a vector valued parameter of the model. We should discuss the families of region models shortly. Therefore the region processes are represented by a vector $W^r$ of unknown dimension.

$$W^r = (K^r, \{(R_i, \ell_i, \Theta_i); \ i = 1, 2, ..., K^r\}). \quad (3.2)$$

We assume that the curve layer $I^c$ is a linear sum of a number of $N$ image bases, following the literature of image coding [55,65,82].

$$I^c = \sum_{j=1}^{N} \alpha_j B_j, \quad B_j \in \Delta, \quad (3.3)$$

$\alpha_j$ is the coefficient and base $B_j$ is selected from an over-complete basis or dictionary $\Delta$, for example, $\Delta$ includes Difference of Gaussians (DoG) and Difference of Offset Gaussians (DOOG) over a group of transforms (scaling, rotating, and translation). These image bases are grouped into a number of $K^c \leq N$ curves $C_i, \ i = 1, ..., K^c$ based on a probabilistic curve model that we should deliberate in section (3.5.1). $C_i$ is a list of bases with certain geometric and photometric regularities. Thus the curve processes are denoted by a vector $W^c$ and $I^c$ is a deterministic function of $W^c$,

$$W^c = (K^c, \{C_i; \ i = 1, 2, ..., K^c\}), \quad I^c = I^c(W^c). \quad (3.4)$$

In a Bayesian framework, our objective is to make inference about $W = (W^r, W^c)$ from $I$ that maximizes a posterior probability,

$$W^* = (W^r, W^c)^* = \arg \max_{\Omega^W} p(W^r, W^c|I) = \arg \max_{\Omega^W} p(I - I^c(W^c)|W^r)p(W^r)p(W^c).$$
In the computation, the region and curve processes $W^r, W^c$ are coordinated by the generative models expressed in equations (3.1) and (3.3), i.e.

$$\mathbf{I}' = \mathbf{I} - \mathbf{I}^c(W^c).$$

In a language of neuroscience, the two layers $W^r, W^c$ (supposed they are represented by two cortical areas) have some mutual inhibition as they compete to explain the observed image, like the lateral inhibition between adjacent neurons (which are bases $B_j$ in our representation). This enables the use of multiple families of image models either 2D or 1D, and distinguishes the generative methods from the discriminative methods for image segmentation (e.g. [54]).

In the following two sections, we discuss the mathematical models for the region and curve processes.

### 3.4 Probabilistic models for region processes

In this section, we briefly overview the probabilistic models for region processes, following the DDMCMC work in [85].

1. **The likelihood for the region layer $\mathbf{I}'$.** This is a product of individual region models,

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

$\ell_i \in \{1, 2, 3, 4\}$ indexes the following four families of intensity models.

**Family 1:** $\omega_1$. This is a simple Gaussian model for flat regions. It assumes that pixel intensities in a region $R$ are constant subject to an independently and identically distributed (i.i.d.) Gaussian noise.

**Family 2:** $\omega_2$. This is a non-parametric model for cluttered regions. It assumes that pixel intensities are iid distributed according to a histogram which is discretized as a step function expressed by a vector $\Theta = (h_1, h_2, \ldots, h_G)$. 

63
Family 3: $\omega_3$. This is Markov random field model (FRAME) for textured regions. We choose a set of 8 filters in FRAME and formulate the model in pseudo-likelihood form.

Family 4: $\omega_4$. This is a spline model for regions with gradually changing intensities, such as lighting areas.

We refer to [85] for detailed specification of these models. In summary, the algorithm can switch between the families to search for a good fit in terms of high likelihood function for each region $R_i$.

2. The prior model for region process $W^r$. The prior model $p(W^r)$ penalizes model complexity and ensure boundary smoothness. Let $A_i = |R_i|$ be the area (pixel number) of $R_i$, and $\Gamma_i = \partial R_i$ be the region boundary, then

$$p(W^r) = p(K^r) \prod_{i=1}^{K^r} p(R_i)p(\ell_i)p(\Theta_i|\ell_i) = p(K^r) \prod_{i=1}^{K^r} p(A_i)p(\Gamma_i)p(\ell_i)p(\Theta_i|\ell_i) 
(3.5)$$

$$p(K^r) \propto e^{-\lambda_0 K^r}, \quad p(A_i) \propto e^{-\gamma A_i}, \quad p(\Gamma_i) \propto e^{-\mu \ell_i}, \quad p(\Theta_i) \propto e^{-\nu \text{len}(\Theta_i)}$$

In these probabilities, $\gamma, c, \nu, \mu$ are some constants and $\text{len}(\Theta_i)$ is the number of parameters in $\Theta_i$.

A DDMCMC algorithm using this set of priors and image models are tested in a large set of images with satisfactory results [85]. It was also tested in a benchmark dataset of 50 natural images by the Berkeley group, and achieved the best results among the algorithms that have been tested. See http://www.cs.berkeley.edu/~martin/segbench/BSDS100/html/benchmark

However, it is evident in these experiments that such region models are not suitable for 1D curve processes. This motivates our curve models below.

3.5 Probabilistic models for curve processes

In this section, we present a rope model for curve processes, and we start with a brief review of existing curve models.

### 3.5.1 A generative rope model of curve processes

In the real images, the curve patterns are generated by elongated objects, such as trees, stems, cables, strings, rails, and so on. Instead of being curves of constant intensities, such objects often have interesting intensity profiles due to lightings and are blended with the background regions through image formation.

Figure 3.3: A rope model of curves consists of a chain of knots. A knot has 1-3 image bases shown by the ellipses.

The curve processes $W_c$ have a number of curves \( \{ C_i; \ i = 1, 2, ..., K_c \} \), and each curve $C$ is a chain of $n$ knots as shown in Figure 3.3.

\[
C = (n, \xi_1, \xi_2, ..., \xi_n)
\]
Each knot $\zeta$ consists of 1-3 image bases. One major base shown by a large ellipse is often associated with 0-2 small minor bases to achieve smooth blending with background regions, thus

$$
\zeta = ( (\alpha_1, B_1), ..., (\alpha_k, B_k) ), \quad k \leq 3.
$$

$\alpha_1, ..., \alpha_k$ are the coefficients of these bases and thus specifies the photometric properties and $B_1, ..., B_k$ are bases selected from an over-complete basis or dictionary $\Delta$. We utilize 134 base functions shown in figure 3.4 which are Gaussians, DoG, and DOOG bases at various orientation and scales, and can be translated to any location.

![Image](image.png)

Figure 3.4: Base functions used for the disctionary $\Delta$.

For each curve $C$ we have a 2nd order Markov chain model for the knots

$$
p(C) = p(n)p(\zeta_1)p(\zeta_2|\zeta_1) \prod_{i=3}^{n} p(\zeta_i|\zeta_{i-1}, \zeta_{i-2}),
$$

where $p(n)$ controls the length of the curve. The conditional probabilities $p(\zeta_2|\zeta_1)$ and $p(\zeta_i|\zeta_{i-1}, \zeta_{i-2})$ are defined so that the curve is smooth in geometry (by energy terms for co-linearity and co-circularity of adjacent bases as in the SNAKE/Elastica models) and in appearance (by energy term for the change of base types and coefficients). This model can be learned by a minimax entropy method as in [96].
We draw a set of random curves (ropes) from the model $p(C)$ in Figure 3.5. Figure 3.5.a shows the geometric curves for the ropes, and the knots of these chains are shown in Figure 3.5.b. Then Figure 3.5.c is the curve layer $I^c$ which is a linear sum of all bases in the rope, as equation (3.3) defines.

![Images](a). b). c).

**Figure 3.5:** Random ropes sampled from the prior model $p(C)$. a). The geometric curves for the sampled ropes. b). A symbolic representation of the sampled ropes. c). Curve layer $I^c$ by the ropes.

To summarize, the curve model for $p(W_c)$ is,

$$p(W_c) = p(K^c) \prod_{i} p(C_i), \quad I^c = I^c(W_c).$$

### 3.6 The solution space $\Omega$

Given the image models in the previous sections, we now briefly analyze the solution space $\Omega$ for $W = (W^r, W^c)$. As $K^r, K^c$ are unknown numbers, the solution space $\Omega$ is

$$\Omega = \left[ \bigcup_{K^r} \Omega^r_{K^r} \right] \times \left[ \bigcup_{K^c} \Omega^c_{K^c} \right],$$

where $\Omega^r_{K^r}$ is the space for $W^r$ with exactly $K^r$ regions and $\Omega^c_{K^c}$ is the space with exactly $K^c$ curves.
In image segmentation, there are two ways of defining a region. One represents a region boundary $\Gamma(s) = \partial R$ as a continuous contour parameterized by $s$ and treats the image domain as a 2D plane. The other considers a segmentation as a label map $\psi_\Lambda$, and a region $R_i$ is a set of pixels sharing the same label, say $n$,

$$R_i = \{(x, y) : \psi(x, y) = n, (x, y) \in \Lambda\}; \text{ for } i = 1, 2, ..., K.$$

In this paper, we adopt the label map notation and define the solution space $\Omega_{K^r}$ as a product of a $K^r$-partition space and $K^r$ spaces for the image models as

$$\Omega_{K^r} = \left[ \omega_{\pi_{K^r}} \times \omega_\Theta \times \cdots \times \omega_\Theta \right]_{K^r},$$

where $\omega_\Theta = \bigcup_{i=1}^4 \omega_{g_i}$. The set of all $K^r$-partitions, $\omega_{\pi_{K^r}}$, is a quotient space of the set of all possible $K^r$-labelings divided by a permutation group, $\mathcal{P}\mathcal{G}$, for the labels. It can be denoted as

$$\omega_{\pi_{K^r}} = \{(R_1, R_2, ..., R_{K^r}) = \pi_{K^r} ; \text{ } |R_i| > 0, \forall i = 1, 2, ..., K^r\}/\mathcal{P}\mathcal{G},$$

where

$$\pi_{K^r} = (R_1, R_2, ..., R_{K^r}), \text{ } \cup_{i=1}^{K^r} R_i = \Lambda, R_i \neq R_j, \forall i \neq j$$

is a set of $K^r$ regions of $\Lambda$.

2. Solution space for $\Omega_{K^e}$

Recall the definition of the curve layer in section 3.5.1. The curve layer is composed of an unknown number of curves; Each curve consists of an unknown number of knots which are made up of 1-3 bases. In correspondence to this definition, the solution space for the
curve layer is defined as

$$\Omega^c_{K^c} = \bigotimes_{K^c} \varpi^c_C.$$  

The space for each curve $$\varpi_C$$ is of the form

$$\varpi_C = \bigcup_{n=0}^{\lfloor |\Lambda| \rfloor} \varpi_{\zeta} \times \ldots \times \varpi_{\zeta},$$

where $$\varpi_{\zeta}$$ is the space for each knot and is defined as

$$\varpi_{\zeta} = \varpi_B \times \varpi_B \times \varpi_B.$$  

In the curve layer, the space for the bases is defined as

$$\varpi_B = \{\phi\} \cup \{(b_i, B_i), i = 1, 2, \ldots, G_b \times N_B \times |\Lambda|\},$$

where $$G_b$$ is the number of possible values for coefficient $$b_i$$, $$N_B$$ is the number of base functions in the over-complete basis and equals to 134 in this paper, and $$|\Lambda|$$ is the size of the lattice.

### 3.7 Integration of region and curve processes by MCMC

Now we turn to the design of the algorithm which forms ergodic Markov chain with reversible jumps [36, 37] and diffusion to explore the solution space $$\Omega$$.  

69
3.7.1 Dynamics design

We use the Metropolis-Hasting algorithm in realizing the jump processes. To achieve the detailed balance equation the acceptance rate is computed as

$$\alpha(W \to dW') = \min(1, \frac{G(W' \to dW)p(W'|I)dW'}{G(W \to dW')p(W|I)dW})$$  \hspace{1cm} (3.6)

where $G(W \to dW')$ and $G(W' \to dW)$ are the two proposal probabilities for the Markov chain to jump between solution $W$ and $W'$. The data-driven techniques are used to propose important proposals to guide the Markov chain in traveling in the solution space $\Omega$ more efficiently. The diffusion dynamics are realized by steepest ascent algorithms.

In the following, we briefly discuss different types of these dynamics for region and curve processes.

**Jump and diffusion dynamics for region processes**

I. **Boundary diffusion** This is a diffusion process used to adjust the region boundaries, which are represented by continuous curves evolving to maximize the posterior probability through a region competition equation [92].

II. **Model adaptation.** This is simply to fit the parameters of a region by steepest descent equation,

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

III and IV. **Split a region into two and Merge two regions into one** These are a pair of reversible jumps. Suppose at a certain time step, a region $R_k$ with model $\Theta_k$ is split into two regions $R_i$ and $R_j$ with models $\Theta_i$, $\Theta_j$, or vice versa, and this realizes a jump between two states $W$ to $W'$.

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

where $W_\sim$ is the remaining variables that are unchanged during the move.
V. **Switch image models.** This switches the image model within the four families for a region $R_i$. For example, from texture description to a spline surface etc.

$$W = (\ell_i, \Theta_i, W_-) \leftrightarrow (\ell_i', \Theta_i', W_-) = W'.$$

**Jump and diffusion dynamics for curve processes**

**VI Curve diffusion.** This is a diffusion process by curve deformation [45].

**VII and VIII. Create a new curve and Delete a curve.** They are a pair of reversible jumps. At a time step, a new curve could be created with 1-3 knots or a curve with less than three knots could be killed. The two solution states before and after the change are denoted as

$$W = (K_C, W_-) \leftrightarrow (K_C + 1, C_{K_C+1}, W_-) = W'. \quad (3.7)$$

**IX and X. Split a curve into two and Merge two curves into one.** They are a pair of reversible jumps as well. At a step, two curves could be merged into a new curve or a curve having at least two knots could be split into two curves. We have the two states defined as

$$W = (K_C, C_i, W_-) \leftrightarrow (K_C + 1, C_i', C_j W_-) = W'. \quad (3.8)$$

There have been a great deal of work done in the area of perceptual organization about how to group 1D elements, such as straight line segments, which potentially belong to the same object together. These grouping methods are considered as bottom-up techniques and could be utilized in designing proposals to help to achieve a fast convergence rate.

**XI and XII. Engage a new knot to a curve and Kill a knot from a curve.** They are two complementary moves. At a step, a new knot is proposed to engage to one of a curve’s two extremes or one of a proposed curve’s two extreme knots could be killed. The two states can be written as

$$W = ((\zeta_1, \ldots, \zeta_n), W_-) \leftrightarrow ((\zeta_1, \ldots, \zeta_n, \zeta_{i+1}), W_-) = W'. \quad (3.9)$$
In the following, we give an example of how to compute the proposal probabilities $G(W \rightarrow dW')$ and $G(W' \rightarrow dW)$ for computing the acceptance rate $\alpha(W \rightarrow dW')$ in equation (3.6). The readers are referred to [85] for a detailed discussion of computing proposal probabilities for region processes.

The two proposal probabilities for move type IX (split a curve into two curves) can be computed as

$$G(W \rightarrow dW') = q(IX)q(C_k)q(C_i|C_k)q(C_j|C_k)dW',$$

and

$$G(W' \rightarrow dW) = q(X)(q(C_i)q(C_j|C_i) + q(C_j)q(C_i|C_j))dW,$$

where $q(IX)$ is the probability for choosing dynamic type IX, $q(C_k)$ is the probability for proposing curve $C_k$ to split. The two new curves, $C_i$ and $C_j$, are created with the probabilities $q(C_i|C_k)$ and $q(C_j|C_k)$ respectively. Similarly, $q(X)$ is the probability for choosing dynamic type X, $q(C_i)$ is the probability for choosing curve $C_i$, and $q(C_j|C_i)$ is the probability for proposing curve $C_j$ to merge with $C_i$. Likewise, we could compute the values for $q(C_j)$ and $q(C_i|C_j)$.

### 3.7.2 Initialization by image pyramid and matching pursuit

To initialize the algorithm, we decompose an image into two layers $I = I_o + I_c$ by a simply Laplacian pyramid method, as Figure 3.6 shows. $I_o$ is a lowpass filtered version of the input image $I$, and $I_c = I - I_o$ is the residual image. In general, $I_o$ contains regions, and $I_c$ contains mostly high frequency components for the curves as well as region boundaries.

Then we adopt a match pursuit method by Mallat and Zhang [55] to quickly compute an initial set of image bases from image $I_o$. In particular, we select the ridge type of bases
Figure 3.6: An input image $\mathbf{I}$ (a) is initially decomposed into two components: $\mathbf{I}'_o$ (b) is a lowpass filtered version, and $\mathbf{I}_c$ is the residue for high frequency components.

with coefficients larger than a certain significant threshold $\tau$.

$$\mathbf{I}_o' = \sum_{\alpha_j \geq \tau} \alpha_j B_j + \mathbf{I}_o'.$$

These selected bases are often good seeds for the curves. The reconstruction residual image $\mathbf{I}_o'$ is added to the region layer $\mathbf{I}_o' \leftarrow \mathbf{I}_o' + \mathbf{I}_o'.

For the region layer $\mathbf{I}_o'$, we conduct some edge detection and clustering as the DDM-CMC algorithm did in [85].

### 3.7.3 Summary of the algorithm

To summarize, we list the algorithm below.

### 3.8 Experiments

We test the algorithm on many natural images and report some results in Figures 3.8, 3.9 and 3.10. Figure 3.8 shows the results for a sea shore image where regions and curves are parsed. Figure 3.8.b shows the regions, and we sample a synthesized image from the likelihood $\mathbf{I}_{syn} \sim p(I|W^r)$ given the computed $W^r$, this illustrates the computed region models. Clearly the shading effects in the water and the sky are captured by the spline model in family 4. Figure 3.8.c is the sketch for the computed curves $W^c$, and Figure 3.8.f is the image $\mathbf{I}^c = \mathbf{I}^c(W^c)$. Thus we have an overall synthesis $\mathbf{I}_{syn} = \mathbf{I}_{syn} + \mathbf{I}^c$ shown in
1. Initialize $I^o = I^e$ and $I^r = I^o$ by the Laplacian pyramid method.

2. Initial $W^e$ by match pursuit, and $W^r$ by clustering.

3. Select a diffusion dynamic or a jump dynamic at random.

4. If a diffusion dynamic is selected, run the following dynamics at random:

   • For type I, a region is randomly proposed to execute region competition for its boundaries.
   • For type II, a region is randomly selected and its corresponding model parameter is adapted by a steepest descent method.
   • For type VI, a curve is randomly selected and a deformation process is executed as in [45].

5. If a jump dynamic is selected,

   • a new solution $W'$ is randomly sampled according to the dynamic picked at random below.
     - For type III, a region is randomly picked to be split into two new regions which are randomly proposed by one of the partition maps.
     - For type IV, two neighboring regions are randomly chosen to form a new region.
     - For type V, a different model is randomly picked for a randomly chosen region.
     - For type VII, 1-3 knots are randomly proposed to form a new curve.
     - For type VIII, a curve is randomly chosen and is then killed.
     - For type IX, a curve is randomly picked and is then split into two new curves.
     - For type X, two neighboring curves are randomly proposed to form a new curve.
       For type XI, a new knot is proposed to engage to one of a randomly picked curve’s two extreme knots.
     - For type XII, one of a randomly picked curve’s two extreme knots is proposed to be killed.

   • The overall posterior probability, $p(W'|I)dW'$, is computed.
   • The proposal probabilities, $G(W' \to dW)$ and $G(W \to dW')$ are then computed.
   • A decision of acceptance is made according the acceptance probability $\alpha(W \to dW')$ in equation 3.6.

6. Repeat the above steps to draw samples $W$ from $p(W|I)$.

Figure 3.7: The algorithm that integrates the region and curve processes.

Figure 3.8.d. By comparing $I^e_{syn}$ and $I^e$ to the initial decomposition $I^e_o$, $I^o$ in Figure 3.6, we clearly see that it is the more sophisticated region and curve models that refine the decomposition. Figure 3.9 shows the segmented regions $W^r$ and $I^e_{syn}$ by the previous DDMCMC algorithms which assumes region processes only.
Figure 3.8: Result: parsing a sea shore image into regions and curves.

Figure 3.9: The segmentation and synthesis with region processes only for fig. 3.8

Figure 3.10 displays more examples in a similar way. The improvement of synthesized images for all the examples demonstrate the the advantages of engaging curve models. The parameters for the two methods are set to be the same to have a fair comparison. Although Figure 3.10.a by the region processes only successfully segmented many tree trunks out, it disconnects the background. The synthesis by region and curve processes for Figure 3.10.b is more similar to the original image than that by the region processes, thus, the new algorithm achieves a better visual effect.
The semantical representation $W$ by region and curve models can largely reduce the coding length of image. The table below lists the number of bytes used by $W$ for the synthesized images in comparison with the jpeg algorithm for the original images.

<table>
<thead>
<tr>
<th>Region and Curve processes</th>
<th>Image a in fig. 3.10</th>
<th>Image b in fig. 3.10</th>
<th>Image c in fig. 3.10</th>
</tr>
</thead>
<tbody>
<tr>
<td>jpeg</td>
<td>2,387</td>
<td>2,266</td>
<td>1,347</td>
</tr>
<tr>
<td>jpeg</td>
<td>17,001</td>
<td>10,620</td>
<td>10,620</td>
</tr>
</tbody>
</table>

Table 3.1: A comparison of coding lengths in bytes for the example images by the proposed algorithm and the jpeg algorithm.

This chapter introduces an additive curve model employed by curve processes in a DDMCMC framework to parse natural images into an unknown number of regions and curves. It demonstrates that: (1) Regions and curves possess different stochastic properties and, thus, it is necessary to separate them into different processes. (2) A DDMCMC algorithm with dynamics that adapt to these properties could be used to parse images into region and curve processes. (3) Some future applications could be seen for image parsing along several directions. The results for the last several examples, however, are not perfect and we could observe some defects such as: (1) broken lines, (2) inflexible shape and intensity values for each curve, (3) small intensity discontinuity and gap between curve knots. The first situation happens because of the lack of bottom-up proposals and dynamic proposals to propose curve knot. The second phenomenon is due to the highly constrained curve knots by image bases which are selected from a limited dictionary of 120 bases. Also because of the limited number of fixed image bases functions, the superposition of these functions will leave some parts of images on the curve not well represented (a typical singularity problem). It is not an easy task to apply diffusion methods to adapt and evolve each curve represented by a chain of base functions. Existing methods concerning the representation of region boundaries [14, 27, 53] and curves such as “wavelets” [55], “bandelets” [69], and “ridgelets” [12] are mostly developed in the context of image compression wherein the problem is tackled as to solve the singularities at edges and curves by projecting them
onto wavelet basis or overcomplete basis, which essential amounts to a function approximation approach. Natural images, however, are created by projecting objects positioned at different palaces in 3D world to 2D plane. The projections of these objects occlude each other in 2D images for complex scenes and generate region boundaries. Besides occlusion, boundaries are also due to lighting, texture, etc. Efforts are made to design “bandelets” and “ridgelets” as coding methods to achieve image compression. However, due to the irregular shape of edges and boundaries, it is not practically feasible and contrary to human perception to parse images of complex scenes directly into a set of fixed functions whose addition approximates the original image. (We could do so for a certain type of images such as some textures.) The defects for the results in the previous chapter also reveal this and demand a better curve model. Many image segmentation methods, either explicitly or implicitly, using a chain of nodes and intensity models to directly code region boundaries and content respectively are found to be very effective. This point is supported by a number of promising results in chapter 2 for image segmentation by DDMCMC. Adopting the ideal of explicit boundary coding and intensity modeling, we propose, in the next chapter, a occlusion curve model for curve processes.
Figure 3.10: More image parsing results by the algorithms with the integration of region and curve processes and by the previous DDMCMC algorithm with region processes only.
CHAPTER 4

Parsing Images into Regions and Occluded Curves

4.1 Introduction

In this chapter, we present an algorithm that parses images into region and curve processes by DDMCMC and make the following contributions.

First, a new generative curve model is presented for modeling curve patterns. Each curve is represented by a function $U(s)$, which is subject to a SNAKE type prior, $H(s)$, width at each position $s$, and an intensity model.

Second, a new diffusion function is developed for evolving curves and the execution of the function is realized in MCMC diffusion dynamics.

Third, we integrate both region and curve models for image parsing and develop an Markov Chain Monte Carlo algorithm to search for global optimization wherein different types of jump and diffusion dynamics are incorporated to explore the complex solution space. Thus, both curves and regions are competing with each other under the same framework in the Bayesian formula.
4.2 Problem formulation in Bayes statistics

Let $\Lambda = \{(i, j) : 1 \leq i \leq L, 1 \leq j \leq H\}$ be an image lattice, and $I$ be an intensity image defined on $\Lambda$. We assume that the image $I$ is composed of two layers,

$$I = I^c \odot I^r,$$

(4.1)

where $I^c$ and $I^r$ are called curve layer and region layer respectively. A small difference between the $I^c$ in this chapter and the one in the previous chapter is that the intensity value of a pixel $v$ in curve layer $I^c_v \in \{\phi, 1, ..., G\}$, which means a pixel could be transparent.

The occlusion operator $\odot$ is denoted as

$$I^c_v \odot I^c_v = \begin{cases} I^r_v & \text{if } I^c_v = \phi \\ I^c_v & \text{otherwise} \end{cases}.$$

Curve layer $I^c$ is assumed to be on top of region layer $I^r$ and the image $I$ is obtained by the superposition of the two layers. Figure 4.1 illustrates an example of $I^c$, $I^r$, and $I$.

Figure 4.1: An example of two-layer image structure.
Region layer $I^r$ consists of a number of $K^r$ disjoint regions which form a partition of the lattice $\Lambda$.

$$\bigcup_{i=1}^{K^r} R_i = \Lambda, \quad R_i \cap R_j = \emptyset, \quad \forall i \neq j.$$  (4.2)

We denote the image intensities in a region by $I^r_R$, and a region is said to be coherent in the sense that $I^r_R$ is a realization from a probabilistic model $p_\ell (I^r_R; \Theta)$. $\ell$ indexes the model or a stochastic process, and $\Theta$ is a vector valued parameter of the model. Therefore, the region processes are represented by a vector $W^r$ of unknown dimension.

$$W^r = (K^r, \{(R_i, \ell_i, \Theta_i); \quad i = 1, 2, ..., K^r\}).$$  (4.3)

We assume that the curve layer $I^c$ consists of a number of $K^c$ curves that may overlap each other. The curve processes are denoted by a vector $W^c$ in the curve layer $I^c$.

$$W^c = (K^c, \{(C_i, \alpha_i); \quad i = 1, 2, ..., K^c\}).$$  (4.4)

The geometric shape of each curve $C_i$ is coded by two non-parametric functions, $U(s)$ and $H(s)$, which respectively account for center line and curve width. The intensity values of each curve are represented by a parametric non-linear function which accounts for its photometric regularity. $\alpha_i$ is the order number of curve $C_i$ among all curves. As in image segmentation, we use the both representations by continuous function for boundary and by lattice labeling to describe the overall shape of a curve. These two representations are considered interchangeable.

Thus, the solution vector is

$$W = (W^r, W^c),$$

81
and we are looking for the optimal solution $W^*$ that maximizes the posterior probability

$$W^* = (W^r, W^c)^* = \arg \max_{W \in \Omega} p(W^r, W^c|I)$$

$$\propto p(I|W^r, W^c)p(W^r, W^c) = p(I|W^c, W^r)p(W^r)p(W^c).$$

(4.5)

We assume that $W^r$ is independent with $W^c$ in this occlusion model. In an additive model, if we’ve known $I^c$ and $I$ already, then $I'$ is uniquely decided by $I - I^c$. However, it is no longer the case in occlusion model because there are many choices of $I'$ since the image parts occluded by $I^c$ are missing data. Without loss of generality, we let $I' = I$ if $I$ and $I^c$ are known.

Next, we explain how the occlusion is applied.

**Definition 1** We define $C_i \cap C_j$ if $\alpha_i < \alpha_j$ which means curve $C_i$ occludes (is on top of) $C_j$.

Let $\Lambda(C_i)$ be the set of all pixels on curve $C_i$. We define a new term

$$\alpha(x, y) = \min\{\alpha_i, (x, y) \in \Lambda(C_i), i = 1..K^c\},$$

and $\alpha(x, y)$ is assigned to 0 if pixel $(x, y)$ is not on any curves. Given the definition of $\alpha$ we are ready to introduce the definition for domain in which curves and regions operate.

**Definition 2** We define the domain of $C_i$ as

$$D(C_i) = \{(x, y) : (x, y) \in \Lambda(C_i) \text{ and } \alpha(x, y) = \alpha_i\},$$

and the domain of $R_i$ as

$$D(R_i) = \{(x, y) : (x, y) \in \Lambda(R_i) \text{ and } \alpha(x, y) = 0\}.$$
Under the above definitions, curves are considered the same as regions at the domain level. Both of them are a set of pixels but with different parametrizations. The union of domains form the complete image lattice and they don’t overlap with each other. Figure 4.2 illustrates the domains for the curves and regions of a solution vector.

\[
\left[ \bigcup_{i=1}^{K'} D(R_i) \right] \cup \left[ \bigcup_{i=1}^{K'} D(C_i) \right] = \Lambda,
\]

\[
D(R_i) \cap D(R_j) = \emptyset, D(C_i) \cap D(C_j) = \emptyset, D(R_i) \cap D(C_j) = \emptyset, \forall i \neq j.
\]

Figure 4.2: An example of curve domains and region domains.

### 4.3 Probabilistic models for region and curve processes

#### 4.3.1 Prior model for region processes

The prior defined for region processes is defined the same as in the image segmentation
\[ p(W^r) = p(K^r) \prod_{i=1}^{K^r} p(R_i|W^c)p(\ell_i)p(\Theta_i|\ell_i) = p(K^r) \prod_{i=1}^{K^r} p_r(A_i)p(\Gamma_i)p(\ell_i)p(\Theta_i|\ell_i) \quad (4.6) \]

\[ p(K^r) \propto e^{-\lambda_0 K^r}, \quad p_r(A_i) \propto e^{-\gamma_r A^0}, \quad p(\Gamma_i) \propto e^{-\mu \int R_i^\ell ds}, \quad p(\Theta_i) \propto e^{-\nu \text{len}(\Theta_i)} \]

except that \( A_i \) amounts to the size of domain \( D(R_i) \). It is computed this way so that regions and curves have fair chance to compete to explain the input image \( I \).

### 4.3.2 Prior model for curve processes

A curve is expressed by two terms

\[ C = (\Gamma, \Theta), \]

where \( \Gamma = (U(s), H(s)), s \in [0, L], U(s) \) is a non-parametric function depicting the continuous centerline (skeleton) of a curve, \( H(s) \) is a function for the width at curve position \( U(s) = (x(s), y(s)) \), \( \Theta \) consists of the parameters for intensity model. Figure 4.3 illustrates how the geometric shape of a curve is decided by the two non-parametric functions.

![Figure 4.3: A generative curve model.](image)
The prior for each curve is

\[ p(C) = p_c(A)p(U)p(H). \] (4.7)

The general idea is that we want to encourage curves to be smooth, not to have dramatic change of width, and not to be too thick.

\[ p_c(A) \propto e^{-\gamma_c A^{0.9}}, \quad p(U) \propto e^{-\int \lambda_1 |\hat{U}(s)|^2 ds}, \quad p(H) \propto e^{-\int \lambda_2 \hat{H}(s)^2 + \lambda_3 \frac{H(s)^3}{s} ds}. \]

\( p_c(A) \), as in the region prior, plays as a role of model calibration and encourages big curves to form. \( A \) is the size of curve domain \( D(C) \) and \( \gamma_c \) is a scale factor. However, the scale factor \( \gamma_c \) in \( p_c(A) \propto e^{-\gamma_c A^{0.9}} \) is set slightly smaller than \( \gamma_r \) in \( p_r(A) \propto e^{-\gamma_r A^{0.9}} \) so that when a large scale of segmentation result is wanted where \( \gamma_r \) and \( \gamma_c \) are big, curves are still encouraged to generate. \( p(U) \) is used for curve smoothness along its developing direction. The first term in \( p(H) \) controls its width change and the second term in \( p(H) \) penalizes thick curves that can better be explained as regions. If the width of a curve is bigger than a threshold, \( d \), then a big penalty is offered.

If we assume independency among all curves then we can arrive at the prior for the curve vector as

\[ p(W^c) = p(K^c) \prod_{i=1}^{K^c} p(C_i), \] (4.8)

where \( p(K^c) \propto e^{-\lambda_c K^c} \).

Figure 4.4 shows some curves sampled subject to this prior.
Figure 4.4: Random curves sampled from the prior model \( p(C) \). a). The geometric curve centerline (skeleton) for the sampled curves. b). Boundaries of the sampled curves. c). Curve image synthesized by the sampled curves.

### 4.3.3 Likelihood model for curve processes

We assume the intensity model of each curve \( C_i \) is described by a parametric function

\[
B_i(x, y) = a_i x^2 + b_i x y + c_i y^2 + d_i x + e_i y + f_i
\]

plus a zero mean gaussian noise. Thus, the likelihood for curve \( C_i \) is

\[
p(I_v(x, y); \Theta_i) = \frac{1}{\sqrt{2\pi}} \exp\left\{ -\frac{(I_v(x,y) - B_i(x,y))^2}{2\sigma_i^2} \right\}.
\]

where \( \Theta_i = (a_i, b_i, c_i, d_i, e_i, f_i, \sigma_i) \) are the parameters and \( \sigma_i^2 \) is the variance.

Under the occlusion model, the overall likelihood is put on the region domains and curve domains as

\[
\log p(I|W^c, W^r) = \sum_{i=1}^{K_c} \int_{D(C_i)} \int D(C_i) \log p(I(x, y); \Theta_i) \, dx \, dy,
\]

\[
+ \sum_{i=1}^{K^r} \int_{D(R_i)} \int D(R_i) \log p_t(I(x, y); \Theta_i) \, dx \, dy
\]

(4.9)
where $p_t(I(x,y); \Theta_t)$ is defined as before in chapter 2. By this definition, each pixel is claimed by either a region or a curve with its likelihood computed accordingly.

### 4.4 The solution space $\Omega$

Given the image models in the previous sections, we now briefly analyze the solution space $\Omega$ for $W = (W^r, W^c)$.

As $K^r$ and $K^c$ are unknown numbers, the solution space $\Omega$ is

$$\Omega = [\cup_{K^r} \Omega_{K^r}^r] \times [\cup_{K^c} \Omega_{K^c}^c],$$

where $\Omega_{K^r}^r$ is the space for $W^r$ with $K^r$ regions and $\Omega_{K^c}^c$ is the space for $W^c$ with $K^c$ curves.

The discussion of solution space for $\Omega_{K^r}^r$ can be seen in section 3.6 and section 2.4.

**Solution space for $\Omega_{K^c}^c$**

The solution space for occlusion map $\alpha$ with a total number of $k$ curves is just a permutation group and can be denoted as

$$\mathcal{U}_{K^c} = \{(1, 2, \ldots, k), (2, 1, \ldots, k), \ldots, (k, \ldots, 2, 1)\}.$$  \hspace{1cm} (4.10)

Same as in the definition of solution space for regions, we adopt the label map notion instead of $U(s)$ and $H(s)$ as continuous functions in defining curve solution space. However, unlike in the region layer where the union of $K^r$ regions form a complete set $\Lambda$ and regions don’t have intersection, curves in curve layer perform independently and could overlap each other. Thus, the solution space for the curve layer is defined as

$$\Omega_{K^c}^c = \underbrace{\mathcal{U}_C \times \ldots \times \mathcal{U}_C}_{K^c} \times \mathcal{U}_{K^c}.$$
The space for each curve $\mathcal{C}$ is of the form

$$\mathcal{C} = \mathcal{S} \times \mathcal{E},$$

(4.11)

where $\mathcal{S}$ includes any subset of $\Lambda$ that is connected, and $\mathcal{E}$ is the space for curve intensity model parameters. The real space in $\mathcal{S}$ we need to visit, apparently, is much smaller due to the way the prior defined which encourages those emerging as “curve” pattern.

### 4.5 Integration of region and curve processes by MCMC

In this section, we design an ergodic and irreducible Markov Chain to search for the optimal solution $W^*$ in the solution $\Omega$ where an input image is parsed into a set of curves and regions.

#### 4.5.1 Dynamics design

For moves among spaces of different dimensions, we again use the Metropolis-Hasting method. To achieve the detailed balance equation, the acceptance rate is computed as

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

(4.12)

where $G(W \rightarrow dW')$ and $G(W' \rightarrow dW)$ are the two proposal probabilities for the Markov chain to jump between solutions $W$ and $W'$. We use diffusion dynamics to realize moves within space of fixed dimensions.

**Jump dynamics**

There are three basic types of jump dynamics: (1) jumps which involve only region vector $W^r$, (2) jumps which involve only curve vector $W^c$, and (3) jumps which involve both region and curves vectors.

**Jump dynamics for region processes only**
Since the change of $W^r$ won’t affect $W^c$, the procedures involved with these jump dynamics, not surprisingly, are the same as before. I and II. Split a region into two and Merge two regions into one. III. Switch image models.

Jump dynamics for curve processes only

![Diagram]

Figure 4.5: An example for the death and birth of a curve.

IV and V. Death and Birth of a curve. These are a pair of reversible jumps. At a time step, a new curve is created probabilistically according to a combination of small curves obtained from bottom-up process. An existing curve also could be proposed at random to be removed. Due to the occlusion structure in which curves are assigned with a sequential number of occlusion order, removing a curve may change the order of all the curves it occludes by one. To make the jumps reversible, a new curve should have non-zero probability to be “inserted” into any position of the existing order and keep the relative occlusion relations of existing curves untouched. The two solution states before and after the change are denoted as

$$W = (K_C, \alpha, W_-) \longleftrightarrow (K_C + 1, C_{K_C+1}, \alpha', W_-) = W'.$$

(4.13)

Figure 4.5 gives an example of such jumps.

The proposal probability for move type IV, birth of a curve, can be computed as
Figure 4.6: “Curvelets” obtained by a matching pursuit method for a sea shore image.

\[ G(W \rightarrow dW') = q(IV)g_{new}(C_{K_C+1}, \alpha')dW', \quad (4.14) \]

where \( q(IV) \) is the probability for choosing dynamic type IV and \( g_{new}(C_{K_C+1}, \alpha') \) is the probability for proposing a new curve \( C_{K_C+1} \) with a new \( \alpha \) map, \( \alpha' \).

Theorem 1 by Mengersen and Tweedie [58] states that a good proposal \( q() \) should be close to the true distribution \( p() \) to achieve fast convergence rate.

\[ p(W'|I) \propto p(C_{K_C+1}, \alpha'|K_C + 1, W_-, I)p(K_C + 1, W_-= I). \quad (4.15) \]

In equation 4.14, \( g_{new}(C_{K_C+1}, \alpha') \) is designed to approximate \( p(C_{K_C+1}, \alpha'|K_C+1, W_-, I) \) as to provide important proposals. We denote \((K_C + 1, W_-, I)\) by \( W_{IV} \) since they are fixed in this move.

Thus,

\[ p(C_{K_C+1}, \alpha'|W_{IV}) = p(\Gamma_{K_C+1}|W_{IV})p(\alpha'|\Gamma_{K_C+1}, W_{IV})p(\Theta_{K_C+1}|\alpha', \Gamma_{K_C+1}, W_{IV}) \quad (4.16) \]
Likewise,

\[ q_{\text{new}}(C_{K_C+1}, \alpha') = q_{\text{new}}(\Gamma_{K_C+1})q_{\text{new}}(\alpha' | \Gamma_{K_C+1})q_{\text{new}}(\Theta_{K_C+1} | \alpha', \Gamma_{K_C+1}). \]  \hspace{1cm} (4.17)

Next, we discuss in detail how to design the proposal probability \( q_{\text{new}}(\Gamma_{K_C+1}) \) in approximating \( p(\Gamma_{K_C+1} | W_{IV}) \) which can be further decomposed as

\[ p(\Gamma_{K_C+1} | W_{IV}) = p(\Gamma_{K_C+1} | K_C + 1, W_-, I) \]
\[ \propto p(I | \Gamma_{K_C+1}, K_C + 1, W_-)p(\Gamma_{K_C+1}, K_C + 1, W_-). \]  \hspace{1cm} (4.18)

By the matching pursuit algorithm, we obtain a set of “metacurves” \( c_i \)

\[ \Delta = \{ c_i : i = 1, 2, ..., n \} \]

and an example can be seen in the right of figure 4.6. The combinations of these metacurves form a solution space for curve shape which can be represented by

\[ \Xi = \{ \bigcup_j c_j : c_j \in \Delta \}. \]

We compute the proposal probability \( q_{\text{new}}(\Gamma_{k+1}) \) for proposing a new curve as follows.

1. A saliency map \( S = \{ S_i : \sum_{i=1}^n S_i = 1 \} \) for metacurves is constructed where \( S_i \) is computed according to the likelihood \( p(I(c_i) | W) \) for the image part on metacurve \( c_i \).

2. A bond value is given to each pair of curves according to an empirical measure of how likely they could be consecutive parts on a curve. (This is computed offline.)

3. A metacurve \( c_i \) is selected at random according to the saliency map and non-zero bonds connected to the \( c_i \) are sampled. A connected component in which \( c_i \) appears is used to propose for the shape of new curve \( C_{K_C+1} \).

The procedures above can be intuitively understood as follows: we want to put a new curve on the place where original image is not represented well by the current solution
$W$. The saliency map is used to reflect the likelihood term $p(\mathbf{I}|\Gamma_{K_c+1}, K_c + 1, W_-)$ in equation 4.18 and the bonds between curve pairs are used to take account into the curve prior $p(\Gamma_{K_c+1}, K_c + 1, W_-)$ in equation 4.18. (We probably can find more efficient ways of looking for candidate curves by using smarter bottom-up techniques.)

To approximate $p(\alpha'|\Gamma_{K_c+1}, W_{TV})$ by $q_{\text{new}}(\alpha'|\Gamma_{K_c+1})$ where $\alpha' \in \mathcal{U}_{K_c+1}$, we fix the linear order in $\alpha$ for existing curves and “insert” the new curve into a random position. The intuitive idea is as follows: Since the size for the space $\mathcal{U}_{K_c+1}$ is $(K_c + 1)!$, we want to avoid sampling directly in this space to have high potential acceptance rate. One way is to preserve the relative order of existing curves and assign the new curve a random one. (Prof. Yuille suggested that we could use some data-driven knowledge such as $T$-junctions to guide sampling the $\alpha$.)

The $p(\Theta_{K_c+1}|\alpha', \Gamma_{K_c+1}, W_{TV})$ is usually very peaky on a single value $\Theta(\Gamma_{K_c+1}, \alpha', W, \mathbf{I})$ which can be directly computed by the least-square fitting method given $(\Gamma_{K_c+1}, \alpha', W, \mathbf{I})$. Thus

$$q_{\text{new}}(\Theta_{K_c+1}|\alpha', \Gamma_{K_c+1}) = \delta(\Theta - \Theta(\Gamma_{K_c+1}, \alpha', W, \mathbf{I})).$$

The proposal probability for move type V, death of a curve, can be computed as

$$G(W' \rightarrow dW) = q(V)q_{\text{delete}}(C_{K_c+1})dW.$$ 

$q(V)$ is the probability for choosing dynamic type V and $q_{\text{delete}}(C_{K_c+1})$ is the probability for choosing curve $C_{K_c+1}$. The true distribution $p(W|\mathbf{I})$ under the condition that only one curve is removed in $W'$ is first approximated

$$\hat{p}(W|\mathbf{I}) \propto \frac{p(W'|\mathbf{I})}{p(C_{K_c+1}, \alpha'|\mathbf{I})} \frac{1}{p(\mathbf{I}|(C_{K_c+1}, \alpha')p(C_{K_c+1})} \propto \frac{p(W'|\mathbf{I})}{p(\mathbf{I}|(C_{K_c+1}, \alpha')p(C_{K_c+1}) (4.19)}$$

92
\( \alpha \) is directly obtained from \( \alpha' \) to preserve the relative order of remaining curves. Thus, \( q_{\text{delete}}(C_{K_C+1}) \) is computed according to the likelihood and prior for curve \( C_{K_C+1} \) to reflect the two terms in equation 4.19. The difference between current and before in choosing a curve is that the curve that has less probability should have high probability to be selected since it is reflected in the denominator.

**VI and VII. Split a curve into two and Merge two curves into one.** These are a pair of reversible jumps as well. At a step, two curves could be merged into a new curve or a curve could be split into two curves. We have the two states defined as

\[
W = (K_C, C_k, \alpha, W_-) \leftrightarrow (K_C + 1, C_i, C_j, \alpha', W_-) = W'.
\] (4.20)

The distribution \( p(W'|I) \) under the condition that \( C_i \) and \( C_j \) are split from \( C_k \) is first approximated by

\[
\hat{p}(W'|I) \propto \frac{p(C_i, C_j, \alpha'|I)}{p(C_k|I)}.
\] (4.21)

The proposal probability for splitting a curve is

\[
G(W \rightarrow dW') = q(VI)q_{\text{split}}(C_k)q_{\text{split}}(C_i, C_j, \alpha|C_k)dW'.
\]

To reflect equation 4.21, \( q_{\text{split}}(C_k) \) is computed so that a curve having big energy on prior is selected, \( C_i \) and \( C_j \) are obtained by cutting probabilistically at the place subject to high energy. \( \alpha' \) is chosen so that the relative order among unchanged curves are untouched and the orders of \( C_i \) and \( C_j \) are randomly chosen.

The distribution \( p(W|I) \) under the condition that \( C_i \) and \( C_j \) are merged into curve \( C_k \) is approximated by

\[
\hat{p}(W'|I) \propto \frac{p(C_k, \alpha|I)}{p(C_i, C_j|I)}.
\] (4.22)
The proposal probability for merging two curves is

\[ G(W' \rightarrow dW) = q(VII)(q_{\text{merge}}(C_i)q(C_j|C_i) + q_{\text{merge}}(C_j)q(C_i, \alpha|C_j))q(\alpha)dW. \]

\(q_{\text{merge}}(C_i)\) is computed according to the number of neighboring curves \(C_i\) has and \(q(C_j|C_i)\) is computed according to local measure between \(C_i\) and \(C_j\).

![Diagram](image)

Figure 4.7: An example of change of curve vector \(W^c\) by splitting a curve into two.

**VIII and IX. Grow a new knot to a curve and Shrink a knot from a curve.** The shape of a curve, in the real implementation, is coded by two lists of nodes that form two plylines (see figure 4.8). We call the part within the area of two consecutive pairs of nodes a knot and a curve is divided into \(n\) number of knots whose length is fixed as a constant. This means parameter \(s\) is discretized into a number of intervals, depending upon the total length of a curve. In such a discretized notion, the growth of curve, and thus the creation of new knots is realized by jumps. Same is true for shrinking curves. Figure 4.8 gives an example of such two jump dynamics.

**X. Swapping orders of two curves.** It is not hard to prove that by exchanging orders one order sequence could change into any other one within a finite number steps. They are picked probabilistically according to the fitness of the curves to the data.

**Jump dynamics for jumps between region and curve process**

**XI and XII. Convert a region into a curve and Convert a curve into a region.** At a certain stage a region may become elongated and thin. It is therefore better to interpret this region
as a curve so that the total energy is decreased. Likewise, a curve may become “big” and “fat” and it is better to represent it as a region.

**Diffusion dynamics**

**Diffusion dynamics for regions**

In section 4.2, we show that regions and curves both operate on their own domains of image lattice under different parametrizations. Thus, boundary diffusions for both regions and curves can still be implemented by the method of region competition. The motion of points $\Gamma_{ij}(s) = (x(s), y(s))$ follows the steepest ascent equation of the log $p(W|I)$ plus a Brownian motion $dB$ along the curve normal direction $\vec{n}(s)$. By variational calculus, this is [92],

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

The first two terms are derived from the prior and likelihood respectively.

**XIII. Boundary diffusion** This is a diffusion process used to adjust the region boundaries, which are represented by continuous curves evolving to maximize the posterior probability through a region competition equation [92] with minor change due to the diffusion function involved with region size and curves. The functions, however, are straight to be obtained.
XV. Model adaptation. This is simply to fit the parameters of a region by steepest descent equation.

**Diffusion dynamics for curves**

XIII. Curve boundary diffusion

This is a dynamics designed to evolve curve boundary to maximize the posterior probability. We shall focus on the energy for each curve first. According to the probability model defined for each curve in section 4.3, we may write the energy for curve $C_i$, $E^c(C_i)$, by three terms, the likelihood, the prior for the size of domain, and the prior of curve shape which are denoted by $E_1(C_i)$, $E_2(C_i)$, and $E_3(C_i)$ respectively.

\[
E^c(C_i) = E_1(C_i) + E_2(C_i) + E_3(C_i)
\]  

(4.23)

where

\[
E_1(C_i) = \int \int_{D(C_i)} -\log p(I(x, y); \Theta_i) dx dy,
\]  

(4.24)

\[
E_2(C_i) = \gamma \left[ \int \int_{D(C_i)} dx dy \right]^{0.9},
\]  

(4.25)

and

\[
E_3(C_i) = \int s \lambda_1 |\dot{U}_i(s)|^2 + \lambda_2 \ddot{H}_i(s)^2 + \lambda_3 \frac{H_i(s)^3}{d^3} ds.
\]  

(4.26)

The diffusion function for curve $C_i$ is

\[
\begin{align*}
\frac{\delta E^c(C_i)}{\delta U} &= \frac{\delta E_1(C_i)}{\delta U} + \frac{\delta E_2(C_i)}{\delta U} + \frac{\delta E_3(C_i)}{\delta U} \\
\frac{\delta E^c(C_i)}{\delta H} &= \frac{\delta E_1(C_i)}{\delta H} + \frac{\delta E_2(C_i)}{\delta H} + \frac{\delta E_3(C_i)}{\delta H}.
\end{align*}
\]  

(4.27)
Boundary diffusion for curve \( C_i \) only involves the change of \( \Gamma_i = (U_i(s), H_i(s)) \). We define the pair points on the left and right side of the curve by \((x_1, y_1)\) and \((x_2, y_2)\), where

\[
x_1 : x_1(U, H) = U_x + \frac{H}{2} \dot{U}_x, \quad y_1 : y_1(U, H) = U_y + \frac{H}{2} \dot{U}_y,
\]

\[
x_2 : x_2(U, H) = U_x - \frac{H}{2} \dot{U}_x, \quad y_2 : y_2(U, H) = U_y - \frac{H}{2} \dot{U}_y.
\]

An illustration can be seen in figure 4.9.

![Figure 4.9: A solution vector.](image)

By Green’s theory we could find function \( L() \) such that

\[
E_1(C_i) = E_1(\Gamma(v)) = \int \int_{D(C_i)} -\log p(I(x, y); \Theta) \, dx \, dy = \int \int_{D(C_i)} f(x, y) \, dx \, dy
\]

\[
= \int_0^1 L(x, \dot{x}, y, \dot{y}) \, dv = \int_0^{0.5} L(x, \dot{x}, y, \dot{y}) \, dv + \int_0^{0.5} L(x, \dot{x}, y, \dot{y}) \, dv
\]

\[
= \int L(x_1, \dot{x}_1, y_1, \dot{y}_1) \, dv - \int L(x_2, \dot{x}_2, y_2, \dot{y}_2) \, dv,
\]

(4.28)
\[
\frac{\delta E_1}{\delta U} = \frac{\partial E_1}{\partial x_1} \frac{\partial x_1}{\partial U} + \frac{\partial E_1}{\partial y_1} \frac{\partial y_1}{\partial U} + \frac{\partial E_1}{\partial x_2} \frac{\partial x_2}{\partial U} + \frac{\partial E_1}{\partial y_2} \frac{\partial y_2}{\partial U} \\
= f(x_1, y_1) \nabla^T (x_1, y_1) \bullet \left( \frac{\partial x_1}{\partial U}, \frac{\partial y_1}{\partial U} \right) - f(x_2, y_2) \nabla^T (x_1, y_1) \bullet \left( \frac{\partial x_2}{\partial U}, \frac{\partial y_2}{\partial U} \right) \tag{4.29}
\]

Likewise,
\[
\frac{\delta E_1}{\delta H} = \frac{\partial E_1}{\partial x_1} \frac{\partial x_1}{\partial H} + \frac{\partial E_1}{\partial y_1} \frac{\partial y_1}{\partial H} + \frac{\partial E_1}{\partial x_2} \frac{\partial x_2}{\partial H} + \frac{\partial E_1}{\partial y_2} \frac{\partial y_2}{\partial H} \\
= f(x_1, y_1) \nabla^T (x_1, y_1) \bullet \left( \frac{\partial x_1}{\partial H}, \frac{\partial y_1}{\partial H} \right) - f(x_2, y_2) \nabla^T (x_1, y_1) \bullet \left( \frac{\partial x_2}{\partial H}, \frac{\partial y_2}{\partial H} \right) \tag{4.30}
\]

\((\frac{\partial E_1}{\partial D}, \frac{\partial y_1}{\partial D}), (\frac{\partial x_2}{\partial D}, \frac{\partial y_2}{\partial D}), (\frac{\partial x_1}{\partial H}, \frac{\partial y_1}{\partial H})\), and \((\frac{\partial x_2}{\partial H}, \frac{\partial y_2}{\partial H})\) are four Jacobian matrices. \(\frac{\delta E_1}{\delta U}\) and \(\frac{\delta E_1}{\delta H}\) have very intuitive interpretations: Statistical forces on both sides of the curve together change the centerline and width of the curve to fit the image better.

Let \(E_2 = \gamma_c F_2^{0.9}\), we could also get
\[
\frac{\delta E_2}{\delta U} = \frac{0.9\gamma_c}{F_2^{0.1}} \left[ \nabla^T (x_1, y_1) \bullet \left( \frac{\partial x_1}{\partial U}, \frac{\partial y_1}{\partial U} \right) - \nabla^T (x_1, y_1) \bullet \left( \frac{\partial x_2}{\partial U}, \frac{\partial y_2}{\partial U} \right) \right] , \tag{4.31}
\]
and
\[
\frac{\delta E_2}{\delta H} = \frac{0.9\gamma_c}{F_2^{0.1}} \left[ \nabla^T (x_1, y_1) \bullet \left( \frac{\partial x_1}{\partial H}, \frac{\partial y_1}{\partial H} \right) - \nabla^T (x_1, y_1) \bullet \left( \frac{\partial x_2}{\partial H}, \frac{\partial y_2}{\partial H} \right) \right] . \tag{4.32}
\]

\(\frac{\delta E_2}{\delta U}\) and \(\frac{\delta E_2}{\delta H}\) act on the curve to move the boundary in favor of big curves.

For the third term in the energy, we have
\[
\frac{\delta E_3}{\delta U} = 2\lambda_1 \kappa \nabla^T (U_x, U_y) , \tag{4.33}
\]
and
\[
\frac{\delta E_3}{\delta H} = 2\lambda_2 H^2 . \tag{4.34}
\]

98
These are the forces acting on the curve to make it smooth and thin.

4.5.2 Summary of the algorithm

To summarize, we list the algorithm below.

1. Select a diffusion or a jump dynamics at random.

2. If a diffusion dynamic is selected, run the following dynamics at random:
   - For type XIII, a region is randomly proposed to execute region competition for its boundaries.
   - For type XV, a region is randomly selected and its corresponding model parameter is adapted by a steepest descent method.
   - For type XIII, a curve is randomly selected and its boundary is diffused according to the diffusion function 4.27 plus a Brownian perturbation.

3. If a jump dynamic is selected,
   - a new solution \( W' \) is randomly sampled according to the dynamic picked at random below.
     - For type I, a region is randomly picked to be split into two new regions which are randomly proposed by one of the partition maps.
     - For type II, two neighboring regions are randomly chosen to form a new region.
     - For type III, a different model is randomly picked for a randomly chosen region.
     - For type IV, a new curve is randomly proposed from “curvelets” by bottom-up process.
     - For type V, a curve is randomly chosen and killed.
     - For type VI, a curve is randomly picked and split into two new curves.
     - For type VII, two neighboring curves are randomly proposed to form a new curve.
     - For type VIII, a new knot is proposed to engage to one of a randomly picked curve’s two extreme knots.
     - For type IX, one of a randomly picked curve’s two extreme knots is proposed to be killed.
     - For type X, two curves are randomly proposed to swap their occlusion orders.
     - For type XI, a region is picked randomly, according to the extent to which it looks like a curve, to convert to a curve.
     - For type XI, a curve is picked randomly, according to the extent to which it looks like a region, to convert to a curve.
• The overall posterior probability, \( p(W'|I)dW' \), is computed.

• The proposal probabilities, \( G(W' \rightarrow dW) \) and \( G(W \rightarrow dW') \) are then computed.

• A decision of acceptance is made according to the acceptance probability \( \alpha(W \rightarrow dW') \) in equation 4.12.

4. Repeat the above steps to draw samples \( W \) from \( p(W|I) \).

4.6 Experiments

We test the algorithm on many natural images and show some results in Figures 4.10, 4.11, and 4.12. For each example, we put the input image together with its parsing result in different aspects into a group. The top-left is the input image. The top-middle shows the region boundaries for the region processes. The top-right is the curve skeletons for the curve processes. By occluding a synthesized curve layer \( I_{syn}^c \sim p(I|W^c) \), shown at the middle of each second row, on a synthesized region layer \( I_{syn}^r \sim p(I|W^r) \), shown at the right of the each row, we can see how a synthesized image resembles the original image. It verifies, in one aspect, the correctness of image parsing results. Thus, regions and curves are competing each other to explain the input images. There are two major parameters to tune, \( \gamma_c \) in equation 4.7 and \( \gamma_r \) in equation 4.6. \( \gamma_r \) amounts to the scale of regions and \( \gamma_c \) is used to control the extent of how favorable curves are against regions. Though they seem to be consistent across all the images, there are slight differences between the images shown in the experiments. For the women, sea shore, fish, leaves, zebra and tree images, the values are set as \( \gamma_r = 4.0, \gamma_c = 3.4, \gamma_r = 8.0, \gamma_c = 5.6, \gamma_r = 7.0, \gamma_c = 5.0, \gamma_r = 7.0, \gamma_c = 5.0 \), \( \gamma_r = 4.0, \gamma_c = 3.4 \), and \( \gamma_r = 8.0, \gamma_c = 6.4 \), respectively.

4.7 Discussions

This chapter is about an algorithm that explores the solution space to interpret input image as an unknown number of regions and curves by MCMC. It introduces a generative occlusion model to deal with curves, which are named “ridgelets” in the wavelet context.
Figure 4.10: Example: parsing images into regions and curves.

Curves, though could be understood as degenerated regions, demonstrate different properties of randomness and, thus, should be characterized by processes different from region processes. Several future applications can be seen for this algorithm such as image compression, non-photo-realistic rendering, object recognition, etc. Natural images contain a number of high level processes in which curves appear to have well organized structures such as parallel curve process, tree process etc. Thus, we should employ high level processes to group and interpret these structures for a better, faster, and more robust understanding of the image contents, which is the goal of image parsing. We also want to use high level
structures such as trees, parallel lines to resolve local ambiguities. Traditional algorithms in object recognition and perceptual organization, very often, use a feed-forward way in which features like edges and corners are extracted first and then are used in grouping. This two-stage procedure suffers the problem of heavily relying on the robustness of the first stage of feature extraction. In the next chapter, we will introduce an algorithm for grouping high level structures by DDMCMC.
Figure 4.11: Example: parsing images into regions and curves.
Figure 4.12: Example: parsing images into regions and curves.
CHAPTER 5

Grouping Curves into Parallel and Tree Structures by MCMC

5.1 Introduction

In this chapter, we discuss a general inference engine which combines the feature extraction stage and grouping stage in the unified image parsing process. Zhu [98] discussed how perceptual organization methods such as Sarkar and Boyer [77] and Geisler [34], are viewed in a generative statistical modeling scheme. In this chapter, we introduce two generative models to model two frequently appearing curve structures, namely, parallel structure and tree structure, and present an algorithm that parses natural images into regions, independent curves, groups of parallel curves, and groups of trees.

5.2 Problem formulation in Bayes statistics

We consider, same as in section 4.2, an image $I = I^c \otimes I^r$ is composed of two layers, $I^c$, the curve layer and, $I^r$, the region layer. We assume that the curve layer $I^c$ consists of $K^c$ independent curves, $K^p$ parallel curve groups, and $K^t$ groups of tree curves. Each group of parallel curves has a sequence of ordered curves which is considered as an Markov chain process. A group of tree curves form a tree structure. Figure 5.1 displays an example of such decomposition.
Figure 5.1: An example of two-layered image structure with independent curves, parallel structures, and trees in the curve layer.

The solution vector now is denoted by four terms

$$W = (W^r, W^c, W^p, W^t),$$

where representation $W^r$ for the region layer $I^r$ is defined in the previous chapter (see in equation 4.3). The representation for the independent curves, $W^c$, is the same as defined before as

$$W^c = (K^c, \{(C_i, \alpha_i); i = 1, ..., K^c\}). \quad (5.1)$$

The representation for the parallel structures

$$W^p = (K^p, \{P_i; i = 1, ..., K^p\}), \quad (5.2)$$

where $P_i$ represents a group of $N^p_i$ parallel curves and

$$P_i = (N^p_i, ((C^p_{ij}, \alpha^p_{ij}); ..., (C^p_{iN^p_i}, \alpha^p_{iN^p_i}))).$$
$\alpha_{ij}^p$ is the occlusion order among all curves including independent curves, curves in parallel structures, and curves in tree structures.

The representation for the tree structures

$$W^t = (K^t, \{T_i; i = 1, ..., K^t\}),$$

(5.3)

where $T_i$ represents a tree and

$$T_i = (N^{t_i}, \{(C_{ij}^t, \alpha_{ij}^t, \zeta_{ij}); j = 1, ..., N^{t_i}\}).$$

where $\zeta_{ij}$ is the index of the parent curve of $C_{ij}^t$.

The task is to design an algorithm searching for the optimal solution $W^*$ that maximizes the posterior probability.\(^8\)

$$W^* = (W^r, W^c, W^p, W^t)^* = \arg \max_{W \in \Omega} p(W^r, W^c, W^p, W^t | \mathbf{I})$$

$$= \arg \max_{W \in \Omega} p(\mathbf{I} | W^r, W^c, W^p, W^t)p(W^r, W^c, W^p, W^t)$$

$$= \arg \max_{W \in \Omega} p(\mathbf{I} | W^r, W^c, W^p, W^t)p(W^r)p(W^c)p(W^p)p(W^t).$$

(5.4)

Next, we explain how the occlusion is applied. Let $\Lambda(C_i)$ be the set of pixels on independent curve $C_i$, $\Lambda(C_{ij}^p)$ be the set of pixels on curve $C_{ij}^p$ in the parallel structures and $\Lambda(C_{ij}^t)$ be the set of pixels on curve $C_{ij}^t$ in the tree structures.

$$\alpha(x, y) = \min \{\min \{\alpha_i, (x, y) \in \Lambda(C_i), i = 1..K^c\},$$

$$\min \{\alpha_{ip}^p, (x, y) \in \Lambda(C_{ij}^p), i = 1..K^p, j = 1..N^{pi}\},$$

$$\min \{\alpha_{it}^t, (x, y) \in \Lambda(C_{ij}^t), i = 1..K^t, j = 1..N^{ti}\}\}$$

(5.5)

\(^8\)Again, we decide not to confuse the reader and just denote $W^r$, $W^c$, $W^p$, $W^t$ as independent although they have dependency in the prior put on the size of their domains.
and \( \alpha(x, y) \) is assigned to 0 if pixel \((x, y)\) is not on any curves.

The domain of \( C_{ij}^p \) is

\[
D(C_{ij}^p) = \{(x, y) : (x, y) \in \Lambda(C_{ij}^p) \text{ and } \alpha(x, y) = \alpha_{ij}^p\},
\]

and the domain of \( D(P_i) \) is

\[
D(P_i) = \cup_j D(C_{ij}^p).
\]

The domain of \( C_{ij}^t \) is

\[
D(C_{ij}^t) = \{(x, y) : (x, y) \in \Lambda(C_{ij}^t) \text{ and } \alpha(x, y) = \alpha_{ij}^p\},
\]

and the domain of \( D(T_i) \) is

\[
D(T_i) = \cup_j D(C_{ij}^t).
\]

The union of domains forms the complete image lattice and they don’t overlap with each other.

\[
[\cup_{i=1}^{K_r} D(R_i)] \cup [\cup_{i=1}^{K_c} D(C_i)] \cup [\cup_{i=1}^{K_p} D(P_i)] \cup [\cup_{i=1}^{K_t} D(T_i)] = \Lambda.
\]

### 5.3 Probabilistic models for region and curve processes

We shall focus on the priors for parallel structures \( W^p \) and tree structures \( W^t \) since the priors on independent curves have been discussed in detail before (see section 4.3.2).

The prior for \( W^p \) is

\[
p(W^p) = \prod_{i=1}^{K_p} p(P_i).
\]
Each $P_i$ is a Markov chain of curves and the prior of which is of the form

$$p(P_i) \propto p(N^{p_i})p_p(A_i)p(C_{ij}^p)\prod_{j=2}^{N^{p_i}}p(C_{ij+1}^p|C_{ij}^p)$$

(5.6)

where $p(N^{p_i}) = e^{-\lambda_0 N^{p_i}}$.

$$p_p(A_i) = e^{-\gamma_p A_i^{\theta^p}}.$$

$A_i$ is the size of domain $D(P_i)$ and $\gamma_p$ is the scale factor for parallel structures.

$p(C_{ij+1}^p|C_{ij}^p)$ is the conditional probability given curve $C_{ij}^p$ being $C_{ij+1}^p$’s previous curve and can be further decomposed into

$$p(C_{ij+1}^p|C_{ij}^p) = p(C_{ij}^p)f^p(C_{ij+1}, C_{ij}^p).$$

$p(C_{ij}^p)$ is similar to the prior defined for an independent curve but does not have the prior on the size of its domain. $f^p(C_{ij+1}, C_{ij}^p)$ is a similarity measure of curve directions, average curve thicknesses, curve lengths, etc. between the two consecutive curves.

The prior defined for $W_t$ is

$$p(W_t) = \prod_{i=1}^{K_t}p(T_i).$$

Each $T_i$ is a Markov tree and the prior of which is of the form

$$p(T_i) \propto p(N^{t_i})p_t(A_i)p(C_{ij}^{t_i})\prod_{j=1}^{N^{t_i}}p(C_{ij+1}^{t_i}|C_{ij}^{t_i})$$

(5.7)

where $p(N^{t_i}) = e^{-\lambda_0 N^{t_i}}$.

$$p_t(A_i) = e^{-\gamma_t A_i^0}. $$

109
$A_i$ is the size of domain $D(T_i)$ and $\gamma$ is the scale factor for trees. $p(C_{ij}^t)$ is similar to the prior defined for an independent curve. $f^t(C_{ij}^t, C_{ij})$ is a measurement between the directions, thicknesses, ending and starting points of curve $C_{ij}^t$ and those of its parent curve $C_{ij}^t$. If $C_{ij}^t$ either continues or is a branch on a curve, it is then encouraged to be $C_{ij}^t$’s parent and is penalized otherwise. Apparently, two parallel should receive high penalty in $f_t$ since they hardly form two consecutive curves on a tree. Figure 5.2 demonstrates a parallel curve group and several trees drawn from the priors defined above.

![Diagram](image)

a). a group of parallel curves  c). three groups of tree curves

Figure 5.2: Random samples of parallel and tree structures drawn from prior $p(W^p)$ and $p(W^t)$.

### 5.4 The solution space $\Omega$

We now briefly discuss the solution space $\Omega$ for $W = (W^r, W^c, W^p, W^t)$.

**Solution space for $\Omega_{N^r}^p$**

To facilitate the definition of solution space for the occlusion map, we emphasize the total number of curves in the parallel structures and tree structures. The space for each parallel curve group $\omega_P$ of $n$ curves is

$$\omega_P^n = \{(C_1, C_2, \ldots, C_n); \ C_i \in \omega_C, i = 1, 2, \ldots, n\},$$
where \( \varpi_C \) is the space for each individual curve and the definition of which can be seen in equation 4.11.

We define the solution space for \( K^n \) parallel groups with total number of \( n \) curves as

\[
\mathcal{T}^n_{K^n} = \{ \varpi_{i_1} \times \ldots \times \varpi_{i_{K^n}} : i_1 + \ldots + i_{K^n} = n \}
\]

Thus, the solution space for parallel groups having total \( n \) curves is

\[
\Omega^n_n = \bigcup_{K^n=1}^{n} \mathcal{T}^n_{K^n}.
\]

**Solution space for \( \Omega^l_{N^t} \)**

The space for each tree structure \( \varpi_P \) of \( m \) curves is

\[
\varpi^m_T = \{(C_i, \zeta^i) ; \; C_i \in \varpi_C, \zeta^i \in [1, m], i = 1, 2, \ldots, m \},
\]

where \( \zeta^i \) is the index of \( C_i \)'s parent curve.

We define the solution space for \( K^t \) tree groups with total number of \( m \) curves as

\[
\mathcal{T}^m_{K^t} = \{ \varpi_{i_1} \times \ldots \times \varpi_{i_{K^t}} : i_1 + \ldots + i_{K^t} = m \}
\]

Thus, the solution space for trees having total \( m \) curves is

\[
\Omega^m_m = \bigcup_{K^t=1}^{m} \mathcal{T}^m_{K^t}.
\]

Thus, we are ready to give the solution space \( \Omega \) for the vector \( W = (W^r, W^c, W^p, W^t) \) as

\[
\Omega = [\bigcup_{K^r} \Omega^r_{K^r}] \times \bigcup_{K^c} \bigcup_{n} \bigcup_{m} [\bigcup_{K^c} \bigcup_{n} \bigcup_{m} [\bigcup_{K^c}^{\alpha \leftarrow n + m}] \times \bigcup_{n} \bigcup_{m} [\bigcup_{K^c}^{\alpha \leftarrow n + m}]
\]

where \( \Omega^r_{K^r} \) is the space for \( W^r \) with exactly \( K^r \) regions and \( \bigcup_{K^c+n+m} \) is the space for occlusion order \( \alpha \) with \( K^c + n + m \) curves (see equation 4.10).
5.5 MCMC design

This section discusses the MCMC design of how to explore the solution space $\Omega$. We will mainly focus on the dynamics that involve grouping parallel curves and grouping tree curves since the problems of region dynamics design and independent curve dynamics design have been discussed extensively in the last several chapters. As we can see in the previous MCMC dynamics, split and merge dynamics play a key role in jumping from one dimension to another in the solution space. This is still the case in dynamics for grouping curves, which demands more efficient methods of making smart split and merge.

**Diffusion dynamics for curve process only**

The structure of a parallel group is decided by how the curves are sequenced in the Markov chain; The structure of a tree is decided by the parent-children relation. We employ jump dynamics to change them. Diffusion function for each independent curve is the same as equation 4.27. However, each curve in a group not only receives forces from other curves, regions, and its own but also gets the force from the generative model that regulates the whole group. This force functions to keep every curve in each group from becoming too “wild” and to stay consistent with other curves.

Recall that the prior for a parallel structure $P_i$ having $N^{pi}$ is

$$p(P_i) \propto p(N^{pi})p_{p}(A_i)p(C_{ij}^p|C_i^p)\prod_{j=2}^{N^{pi}} p(C_{ij+1}^p|C_{ij}^p).$$

We add the likelihood and define the energy for $P_i$ as

$$E^p(P_i) = E_1(P_i) + E_2(P_i) + E_3(P_i) + \lambda_0 N^{pi}$$

$E_1(P_i)$ is the likelihood and is of the form

$$E_1(P_i) = \sum_{j=1}^{N^{pi}} \int \int_{D(C_{ij}^p)} -\log p(I(x, y); \Theta_{ij})dxdy. \quad (5.8)$$
$E_2(P_i)$ is the energy on the size of the domain

$$E_2(P_i) = \gamma_p \left[ \sum_{j=1}^{N_{pi}} \int \int_{D(C_{ij}^t)} dxdy \right]^{0.9} \tag{5.9}$$

$E_3(P_i)$ is the energy on the shape of each curve

$$E_3(P_i) = \sum_{j=1}^{N_{pi}} E_3(C_{ij}) \tag{5.10}$$

where $E_3(C_{ij}^p)$ is the energy on the shape curve $C_{ij}^p$ which is defined in equation 4.26.

$E_4(P_i)$ is the energy on the consecutive curves

$$E_4(P_i) = \sum_{j=1}^{N_{pi}} f^p(C_{ij+1}^p, C_{ij}^p). \tag{5.11}$$

Thus, the diffusion functions for each curve $C_{ij}^p$ in parallel structure is

$$\begin{align*}
\frac{\delta E^p(P_i)}{\delta U_{ij}} &= \frac{\delta E_{U_j}}{\delta U_{ij}} + \frac{\delta E_{U_i}}{\delta U_{ij}} + \frac{\delta E_{U_i}}{\delta U_{ij}} + \frac{\delta E_{U_j}}{\delta U_{ij}} \\
\frac{\delta E^p(P_i)}{\delta H_{ij}} &= \frac{\delta E_{H_j}}{\delta H_{ij}} + \frac{\delta E_{H_i}}{\delta H_{ij}} + \frac{\delta E_{H_i}}{\delta H_{ij}} + \frac{\delta E_{H_j}}{\delta H_{ij}} \tag{5.12}
\end{align*}$$

Recall that the prior for a tree structure $T_i$ having $N^t_i$ is

$$p(T_i) \propto p(N^t_i)p_e(A_i) \prod_{j=1}^{N^t_i} p(C_{ij}^t) f^t(C_{ij}^t, C_{ij}^t)$$

We add the likelihood and define the energy for $P_i$ as

$$E^t(T_i) = E_1(T_i) + E_2(T_i) + E_3(T_i) + \lambda_0 N^t_i$$
$E_1(T_i)$ is the likelihood and is of the form

$$E_1(T_i) = \sum_{j=1}^{N_{ti}} \int \int_{D(C_{ij}^t)} -\log p(I(x,y) ; \Theta_{ij}) dx dy. \quad (5.13)$$

$E_2(T_i)$ is the energy on the size of the domain

$$E_2(T_i) = \gamma_p \left[ \sum_{j=1}^{N_{ti}} \int \int_{D(C_{ij}^t)} dx dy \right]^{0.9} \quad (5.14)$$

$E_3(T_i)$ is the energy on the shape of each curve

$$E_3(T_i) = \sum_{j=1}^{N_{ti}} E_3(C_{ij}^t) \quad (5.15)$$

where $E_3(C_{ij}^t)$ is the energy on the shape curve $C_{ij}^t$ which is defined in equation 4.26.

$E_4(T_i)$ is the energy on the consecutive curves

$$E_4(T_i) = \sum_{j=1}^{N_{ti}} f^t(C_{ij+1}^t, C_{ij}^t). \quad (5.16)$$

Thus, the diffusion functions for each curve $C_{ij}^t$ in tree structure is

$$\begin{align*}
\frac{\delta E^t(T_i)}{\delta U_{ij}} &= \delta E_1 + \delta E_2 + \delta E_3 + \delta E_4 \\
\frac{\delta E^t(T_i)}{\delta H_{ij}} &= \delta E_1 + \delta E_2 + \delta E_3 + \delta E_4
\end{align*} \quad (5.17)$$

### 5.5.1 A generalized graph labeling method

We argued earlier in this chapter that there is a need for efficient algorithm to make split and merge. To speed up Gibbs sampler [35], Swendsen and Wang [83] proposed a well celebrated algorithm by introducing auxiliary variables for rapid change of states. The algorithm was orginaly proposed to ameliorate the convergence speed of Gibbs sampler over the Potts model. The idea is as follows: It usually takes exponentially waiting time to
change the label of a set of neighboring sites from one to another since the label needs to be updated one by one. For each update, the energy almost remains the same. Swendsen and Wang introduced bond variables between every pair of neighboring sites and use them to indicate, in each step, whether the neighboring sites should be sampled with the same label. By doing so, each connected component linked through the bonds form a cluster and is sampled with the same label independently with other connected components. This dramatically improves the sampling speed especially when we want to change the label for a group of sites from one to another. Higdon [39] later found that partially release the dependency between connected components helps to regulate the change of the label of each connected component. This is done by putting some constraints given by the neighboring components instead of treating them completely independent. A so called partial decoupling algorithm by Higdon [39] is then introduced to release complete independency between bonded sets of sites. There are various other labeling algorithms [9, 41, 80, 88] proposed during the past years tackling the problem of graph labeling and cuts problem. Barbu and Zhu [4] recently discovered a new generalized graph labeling/cuts algorithm that achieves fast convergence rate for general energy functions. The main idea is along the vein of using auxiliary bond variables and data-driven information to help Markov chain walking in the high dimension space. Figure 5.3 illustrates the idea and the algorithm works as follows:

1. For all the sites, not necessarily seated in a regular lattice, that are labeled with a same number, establish bonds connecting neighboring sites whose strengths are the empirical probability of how likely the neighboring sites will have the same label. The empirical probability usually is obtained from data input. (Bottom-up techniques are used.).

2. Sample all the bonds as either $P_{on}/P_{off}$ according to the empirical probabilities.

3. Randomly pick up a connected component and randomly propose a label for all the sites in the component.
4. Use the Metropolis-Hasting algorithm to check whether or not this jump is accepted.

5. Resample the bond variables to form new components.

Figure 5.3: A generalized graph labeling methods. The top left figure shows a status \( W \) where different sets of sites are labeled with different values shown by different shadings. The neighboring sites with the same label is assigned a bond indicating how likely the pair should be bonded. We then sample these bonds \( P_{on}/P_{off} \) in the top right graph to obtain a intermediate graph in which sites are separated into several connected components. From this graph, we randomly sample one of the connected component with a new label. The top right figure displays the new graph where edges are updated according to the new status \( W' \). The same procedures can be applied if we want go from the status shown in the top right figure to the status shown in the top left figure. The bottom figure demonstrates the same stage into which both \( W \) and \( W' \) enter with one to one correspondence as the reason why the cancellation works.

The underlying idea of this algorithm is two-folds: (1) Local image features are used to help bonding similar sites. (2) Heavy burden of computation of all possible proposals leading to the same status change is eased by cancellation. We offer a simple explanation
here and a detailed discussion and proof can be seen in [4]. The critical thing in Metropolis-Hasting algorithm is to compute
\[ \frac{G(W' \rightarrow W)p(W')}{G(W \rightarrow W')p(W')} \]

We could arrive at the conclusion
\[ \frac{G(W' \rightarrow W)}{G(W \rightarrow W')} = \frac{\prod_{i \in g, j \in f} (1 - p(e_{ij}))q(l_A)}{\prod_{i \in g, j \in h} (1 - p(e_{ij}))q(l_B)} \]

where \( p(e_{ij}) \) is empirical probability of sites \( i \) and \( j \), \( q(l_A) \) is the probability of proposing label \( l_A \), and \( q(l_B) \) is the probability of proposing label \( l_B \). The intuitive idea could also be seen in figure 5.3. Once all the edges between \( g \) and \( h \) and edges between \( g \) and \( f \) are removed, and the label of component \( g \) becomes undecided, state \( W \) and \( W' \) enter the same state \( C \) in which all possible choices become identical.

**Jump dynamics for parallel curve groups**

1. **Split-merge parallel curve group** The above method allows us to combine the two dynamics, split and merge, into one dynamics which we call a split-merge dynamics. One problem associated with it, however, is that all the bonds have to be sampled and one connected component is selected probabilistically among all the connected components. This could be a heavy computation when the volume of graph becomes big. Having observed that the selected connected component may have nothing to do with the rest sampled bonds of other labels, we could change the first step in the above algorithm by: (1) Randomly pick up one node with equal probability among all the nodes in the graph. (2) Sample all the bonds among the nodes with the same label as the selected one. This way, we don’t need to sample all other bonds while being able to keep the elegance of cancellation. To facilitate the usage of this new generalized graph labeling method, we extend our notation in the curve vector \( W^c \) in equation 5.1 by considering three types of groups, namely, group I in which there is only one curve, group II in which there are a number of parallel curves, group III in which there are tree curves. Thus, every curve has to be in a group, either by its
own as an independent one or in a parallel or tree group. The *split-merge dynamics* works as follows:

1. Randomly select a curve among all the curves that are in a parallel curve group with equal probability.

2. Sample all the bonds of a group which has the selected curve.

3. Choose a connected component at random.

4. Randomly propose to either let the chosen component to be another parallel curve group or to merge with a neighboring parallel curve group.

5. Compute all the probabilities necessary to decide whether or not this jump is accepted.

6. Rebuild bonds in the new group if a jump is accepted or keep the old status otherwise.

Figure 5.4: An example of split-merge parallel curve group.

There are still some other differences between giving a label to a connected component and choosing a model for a connected component. In the former case, each group of sites is
implicitly denoted by gathering all the sites with the same label. In the latter case, however, each group is explicitly represented by denoting a set of sites with which a model type and parameters are associated. Thus, we can’t abuse the use of the algorithm [4] and special care must be given to make sure that both the reversibility and cancellation are guaranteed. Figure 5.4 gives an example of such moves. Even many restrictions have been put in the procedure, we must pay some attentions in computing the $G(A \rightarrow B)$ and $G(B \rightarrow A)$. For example, from status $A$, the selection of component $d$ or $c$ will both arrive at the same final state $B$.

II and III Grow a parallel curve group and Shrink a parallel group Notice that jump dynamics I can not attach a curve in group I to a parallel curve group since, in step 2, the neighboring group has to be a parallel curve group. They are a pair of dynamics. For dynamics II, a parallel curve group is randomly proposed to which a curve of group I is proposed to attach. For dynamics III, a parallel curve group is randomly proposed in which one curve is proposed to be removed from this group and form a group I by its own.

IV and V Create a new parallel curve group and Disperse a parallel curve group We can also observe that the algorithm by Barbu and Zhu can not change the labels of a set of sites with different labels into one, which is of critical importance in perceptual grouping. Dynamics IV and V are a pair of jumps dealing with this situation. In dynamics IV, existing curves are clustered according to their lengths, directions, and widths into several clusters. One cluster is then proposed and each curve is proposed probabilistically according the its closeness to the mean of the cluster. A new parallel curve group is created with all proposed curves. In dynamics V, a parallel curve group is randomly chosen and all curves are dismissed into group I, individual curve.

VI and VII Attach a new curve to a parallel curve group and delete a curve from parallel curve group With the global structure of a parallel group, we are well advised to “predict” a new curve which may even have nearly zero probability to be proposed from bottom-up processes. Thus, we are released, to a certain degree, from purely relying on proposals by data-driven techniques. Dynamics VI is designed to implement this jump. Dynamics VII,
as a counterpart of dynamics VI, is used to delete a curve directly from a parallel curve group.

**Jump dynamics for tree curve groups**

**VIII and IX Split a tree curve group and Merge two tree curve groups** In the case of split/merge for tree curve groups, how to design smart jumps for splitting and merging trees doesn’t seem to be so critical since every tree has a well established structure wherein each curve has at most one parent curve and several children curves. Removal of a curve from a tree could just break a tree into two. In dynamics VIII, a tree curve group is randomly proposed in which a curve is chosen at random to be removed from this tree. The chosen curve is then set as the root of a new tree and all the parent-children relationships remain untouched. In dynamics IX, two trees are randomly proposed to form a new tree. Figure 5.5 displays an example of spitting and merging tree curve groups.

![Diagram](image)

Figure 5.5: An example of splitting and merging tree curve groups.

**X and XI Grow a tree curve group and Shrink a tree group** They are a pair of dynamics. For dynamics X, a parallel curve group is randomly proposed to which a curve of group I is proposed to attach as a leaf curve. A leaf is defined as one that doesn’t have any child
curves. For dynamics XI, a tree curve group is randomly proposed in which one leaf curve is proposed to be removed from this group and form a group I by its own.

XII and XIII Create a new tree curve group and Disperse a tree curve group Unlike dynamics IV, it is not a easy task to cluster individual curves into clusters of trees. Inspired by the idea of using bond edges, we can similarly establish another graph which is equipped with bond linking individual curves by how likely they form a parent-child relation in a tree. In dynamics XII, each bond is sampled and a connected component is selected. The number of total curves in a component is limited to four so that it won’t be too hard to compute the proposal probability since we can no longer apply the cancellation method here. In dynamics XIII, a tree curve group of less than four curves is selected to be dispersed into individual curves. Figure 5.6 shows an example of such a bond graph in the cue of parent-child relationship for tree group.

![Diagram](image)

Figure 5.6: An example of splitting and merging tree curve groups.

XIII and XV Attach a new curve to a tree curve group and delete a curve from a tree curve group With the global structure of a tree group, we are well advised to “predict” a new curve. Dynamics XIII is designed to implement this jump. Dynamics VII, as a counterpart of dynamics VI, is used to delete a curve directly from a tree curve group.

XVI Change a tree structure A tree group is proposed in which a curve is randomly proposed to be the root of this tree.

Jump dynamics for parallel and tree curve groups

121
XVII Switch group type For any group of one of three types, it is proposed to switch among them. A group of more than one curves can not be proposed to be group I.

5.5.2 Summary of the algorithm

We combine, as usual, all the above dynamics plus dynamics discussed in the previous chapters for region and individual curve processes, and propose them at random to explore the complex solution space looking for the $W^*$ that best “interprets” input images under our models.

5.6 Experiments

We test the algorithm on many natural images and report some results in Figures 5.7, 5.8, 5.9, and 5.10. For each example, we put the input image together with its parsing result into a group. The synthesized curve groups are shown at the top of each group.

This chapter is about an algorithm that explores the solution space to interpret input images as an unknown number of regions and curve groups by MCMC. Thus, regions, individual curves and curve groups are competing each other to explain the input images. There are four major parameters to tune, $\gamma_c$ in equation 4.7 and $\gamma_r$ in equation 4.6, $\gamma_p$ in equation 5.6, and $\gamma_t$ in equation 5.6. They are used to control the degree of how favorable individual curves, parallel curve groups, tree curve groups are against regions. Though they seem to be consistent across all the images, there are some slight differences for the images shown in the experiments. It usually takes 30-60 minutes to run the program for each image.
Figure 5.7: Result: parsing a sea shore image into regions and curves.
Figure 5.8: Result: parsing a sea shore image into regions and curves.
Figure 5.9: Result: parsing a sea shore image into regions and curves.
Figure 5.10: Result: parsing a sea shore image into regions and curves.
CHAPTER 6

Analysis of DDMCMC Convergence Rate

This part includes some technical details for studying the convergence rate of DDMCMC, which could be analyzed by solving three sub-problems.

1). What are the upper and lower bounds of the convergence rate for an irreducible and aperiodic Markov Chain, mostly reversible, with transition matrix $P$? It is desirable to have tight bounds w.r.t. the conductance of $P$ through which the efficiency of Markov Chains is tied with some meaningful design criterion of $P$. The efficiency here is referred to the number of steps of a Markov Chain to reach its stationary distribution starting from an arbitrary distribution. We will show, in the sense of bounds, that the higher the conductance of $P$ is the more efficient the corresponding Markov Chain is by proving two bounds of the second largest eigenvalue modulus in terms of the conductance. This concludes that we should construct matrix $P$ with high conductance to have good design, regarding to convergence rate, of Markov Chains.

2). What are the criterion that one could use to design high conductance transition matrix $P$ for efficient Markov Chains? The criterion that we derive should be intuitive, easy to follow and linked directly to the reason of using data driven techniques in guiding jump dynamics.

3). Can we analytically compute the conductance of the transition matrix $P$ of the data-driven Markov Chains such as those designed in DDMCMC segmentation algorithms? We shall show at least that how different proposal distribution $q$ used in Metroplis-Hasting algorithm affect the overall conductance.
If we can answer the above three questions then we will be convinced that why the three algorithms we developed, 1) Rong Zhang’s object recognition 2) Qiming Luo’s 1D signal segmentation 3) grey scale and color image segmentation, by data-driven MCMC could drastically improve the efficiency of general MCMC algorithms and we should be able to give the convergence rates of our algorithms by computing the conductance analytically.

6.1 Bounds of convergence rate

**Perron-Frobenius Theorem**

Let A be a nonnegative irreducible and aperiodic \( r \times r \) matrix. There exists a real eigenvalue \( \lambda_1 \) with algebraic as well as geometric multiplicity one such that \( \lambda_1 > 0 \), and \( \lambda_1 > |\lambda_j| \) for any other eigenvalue \( \lambda_j \). Moreover, the left eigenvector \( u_1 \) and the right eigenvector \( v_1 \) associated with \( \lambda_1 \) can be chosen positive and such the \( u_1^Tv_1 = 1 \). Let \( \lambda_2, \lambda_3, ..., \lambda_r \) be the eigenvalues of A other than \( \lambda_1 \) ordered in such a way that \( \lambda_1 > |\lambda_2| \geq \cdots \geq |\lambda_r| \) and if \( |\lambda_2| = |\lambda_j| \) for some \( j \geq 3 \), then \( m_2 \geq m_j \), where \( m_j \) is the algebraic multiplicity \( \lambda_j \). Then

\[
A^n = \lambda_1^n v_1 u_1^T + O(n^{m_2-1}|\lambda_2|^n).
\]

**Comments:**

1). If A is stochastic, the \( \lambda_1 = 1 \).

2). If A is stochastic but not irreducible, then the algebraic and geometric multiplicities of the eigenvalue 1 are equal to the number of communication classes.

3). If A is stochastic and irreducible with period \( d > 1 \), then there are exactly \( d \) distinct eigenvalues of modulus 1, namely the \( d \text{th} \) roots of unity, and all other eigenvalues have modulus strictly less than 1.

The proof of convergence rate starts from Perron-Frobenius theorem. Besides the conditions required by this theorem for A, the transition matrix \( P \) we are analyzing is, fortunately, stochastic and reversible. We are much released by this assumption and shall focus
on the second largest eigenvalue modulus, $\lambda_2$, which decides the asymptotical behavior of the Markov Chain. Some of the definitions in this chapter are due to Bremaud 1999 [11].

**Definition 1**

To facilitate the proofs it will be easier to have some definitions defined in $\ell^2(\pi)$ space.

The scalar product between two vectors $x$ and $y$ is

$$\langle x, y \rangle_\pi = \sum_{i \in E} x(i) y(i) \pi(i),$$

where $E$ is a finite state space, $\{1, 2, \ldots, r\}$.

The norm of vector $x$ is

$$\|x\|_\pi = \left( \sum_{i \in E} x(i)^2 \pi(i) \right)^{\frac{1}{2}}.$$

We could write the mean as

$$\langle x \rangle_\pi = \langle x, 1 \rangle_\pi.$$

The variance is then

$$Var_\pi(x) = \|x\|_\pi^2 - \langle x \rangle_\pi^2.$$

**Definition 2**

The *Dirichlet form* $\varepsilon_\pi(x, x)$ associated with a reversible pair $(P, \pi)$ is defined by

$$\varepsilon_\pi(x, x) = \langle (I - P)x, x \rangle_\pi.$$

**Rayleigh’s Theorem**
Let $P$ be an irreducible transition matrix on a finite state space, and let $\pi$ be its stationary distribution. If $(P, \pi)$ is reversible, then for $j \geq 2$,

$$1 - \lambda_j = \inf \left\{ \frac{\varepsilon(x, x)}{\Var(\pi)} : \langle x, v_i \rangle = 0 \text{ for } i \in [1, j-1], \ x \neq 0 \right\},$$

where $1 = \lambda_1 > \lambda_2 \geq \cdots \geq \lambda_r \geq -1$.

**Proposition 1** Let $P$ be the transition matrix of an irreducible and aperiodic Markov Chain and be reversible w.r.t. the stationary distribution $\pi$. We denote $\tilde{P} = D^{-1}P^TD$, where $D$ is a diagonal matrix with $(\pi(i), i = 1..r)$ on the diagonal. Then matrix $M = P\tilde{P}$ is reversible and has nonnegative eigenvalues only. Further, we can have a simplified version of $M$ as

$$M = PP.$$

**Proof.** We could write

$$\tilde{P}_{ij} = \pi(i)^{-1}\pi(j)P_{ji}$$

by the definition. Thus

$$M_{ij} = \sum_{k=1}^{r} P_{ik}\tilde{P}_{kj} = \sum_{k=1}^{r} \pi(k)^{-1}\pi(j)P_{ik}P_{jk} = \sum_{k=1}^{r} \pi(k)^{-1}\pi(k)P_{ik}P_{kj} = \sum_{k=1}^{r} P_{ik}P_{kj}$$

due to the reversibility of $P$, $\pi(i)P_{ij} = \pi(j)P_{ji}$. Therefore we have proven

$$M = PP. \quad (6.1)$$

130
The reversibility of $M$ could be seen as

$$
\pi_i M_{ij} = \pi_i \sum_{k=1}^{r} P_{ik} P_{kj} = \sum_{k=1}^{r} \pi_i P_{ik} P_{kj}
= \sum_{k=1}^{r} \pi_k P_{ki} P_{kj} = \sum_{k=1}^{r} \pi_j P_{ki} P_{jk}
= \pi_j M_{ji}.
$$

Observe that $M$ has the same eigenvalues as $D^{\frac{1}{2}}MD^{-\frac{1}{2}}$, which could be rewritten as a symmetric definite nonnegative matrix with all eigenvalues belonging to the interval $[0, 1]$, $(D^{\frac{1}{2}}PD^{-\frac{1}{2}})(D^{\frac{1}{2}}PD^{-\frac{1}{2}})^T$.

**Proposition 2** Let $P$ be the transition matrix of an irreducible and aperiodic Markov Chain and be reversible w.r.t. the stationary distribution $\pi$. Let $M = PP$ then the convergence rate is determined by the SLE of $M$, $\lambda_2$.

**Proof.** If $n = 2 \times k$, then using **Perron-Frobenius theorem**

$$
P^n = 1\pi^T + O(\lambda_2(M)^k),
$$

where $\lambda_2(M)$ is the SLE of matrix $M$. If $n = 2 \times k + 1$, then using **Perron-Frobenius Theorem**

$$
P^n = 1\pi^T + O(|\lambda_2(P)|\lambda_2(M)^k),
$$

where $\lambda_2(P)$ is the SLEM of matrix $P$. Thus we have

$$
P^n = 1\pi^T + O(\lambda_2(M)^{\frac{1}{2}k}),
$$

\[\square\]
Instead of focusing on the SLEM(second largest eigenvalue modulus) of \( P \), we are much released by studying the SLE(second largest eigenvalue) of \( M \), which is easier to analyze and is identical to SLEM for \( M \).

**Definition 3** For a nonempty set \( B \subset E = 1, \ldots, r \), one defines the capacity of \( B \),

\[
\pi(B) = \sum_{i \in B} \pi(i),
\]

and the ergodic flow out of \( B \),

\[
F(B) = \sum_{i \in B, j \notin B} \pi(i)P_{ij}.
\]

Define for \( B \) not empty,

\[
\psi(B, P) = \frac{F(B)}{\pi(B)}.
\]

The conductance of the pair \((P, \pi)\) is

\[
\psi(P) = \inf \left\{ \psi(B, P); 0 < |B| < r, \pi(B) \leq \frac{1}{2} \right\}.
\]

**Conductance Bounds** The SLE of matrix \( M \) is bounded by its conductance \( \psi(M) \) as

\[
1 - 2\psi(M) \leq \lambda_2 \leq 1 - \frac{\psi(M)^2}{2}.
\]  

**Proof.** (Most of it is due to Bremaud 1999 [11])

(a) We start our proof from the left inequality first. Let \( u \) be a left eigenvector of \( P \) associated with an eigenvalue \( \lambda \neq 1 \). in particular, \( u \) is orthogonal to \( \pi \), the left eigenvector associated with \( \lambda_1 \), and therefore \( u \) has positive as well as negative entries. Assume \( S = \)
1, ..., k, k ∈ [1, r] that u(i) > 0, i ∈ S and π(S) ≤ \frac{1}{2}. Define

\[ y(i) = \begin{cases} 
\frac{u(i)}{π(i)} & \text{if } u(i) > 0 \\
0 & \text{otherwise}
\end{cases} \]

We have

\[ u^T(I - M) = u^T(1 - λ), \]

and therefore

\[ u^T(I - M)y = (1 - λ)u^Ty = (1 - λ) \sum_{i \in S} π(i)y(i)^2. \]

Also,

\[ u^T(I - M)y = \sum_{i \in S} \sum_{j=1}^{r} (δ_{ji} - M_{ji})u(j)y(i) \]

\[ ≥ \sum_{i \in S} \sum_{j \in S} (δ_{ji} - M_{ji})u(j)y(i) \]

since the missing terms −M_{ji}u(j)y(i) corresponding to i ∈ S and j /∈ S are negative or null. Therefore,

\[ u^T(I - M)y ≥ \langle y, (I - M)y \rangle_π, \]

and because

\[ \langle (I-M)x, x \rangle_π = \frac{1}{2} \sum_{i,j \in B} π(i)M_{ij}(x(j) - x(i))^2, \]

we see

\[ 1 - λ ≥ \frac{\sum_{i<j} π(i)M_{ij}(y(i) - y(j))^2}{\sum_{i \in S} π(i)y(i)^2}. \]
From \((a + b)^2 \leq 2(a^2 + b^2)\), we obtain

\[
\sum_{i<j} \pi(i) M_{ij} (y(i) + y(j))^2 \leq 2 \sum_{i<j} \pi(i) M_{ij} (y(i)^2 + y(j)^2),
\]

and, by reversibility,

\[
\sum_{i<j} \pi(i) M_{ij} (y(i)^2 + y(j)^2) = \sum_{i<j} \pi(i) M_{ij} y(i)^2 + \sum_{i<j} \pi(j) M_{ji} y(j)^2
\]

\[
= \sum_{i \neq j} \pi(i) M_{ij} y(i)^2
\]

\[
\leq \sum_{i \in S} \pi(i) y(i)^2.
\]

Thus

\[
1 - \lambda \geq \left( \frac{\sum_{i<j} \pi(i) M_{ij} (y(i) - y(j))^2}{\sum_{i \in S} \pi(i) y(i)^2} \right) \left( \frac{\sum_{i<j} \pi(i) M_{ij} (y(i) + y(j))^2}{2 \sum_{i \in S} \pi(i) y(i)^2} \right).
\]

By the Cauchy-Schwarz inequality

\[
\left( \sum_{i<j} \pi(i) M_{ij} (y(i)^2 - y(j)^2) \right)^2 \leq \left( \sum_{i<j} \pi(i) M_{ij} (y(i) - y(j))^2 \right) \left( \sum_{i<j} \pi(i) M_{ij} (y(i) + y(j))^2 \right),
\]

and therefore

\[
1 - \lambda \geq \frac{1}{2} \left( \frac{\sum_{i<j} \pi(i) M_{ij} (y(i)^2 - y(j)^2)}{\sum_{i \in S} \pi(i) y(i)^2} \right)^2
\]

134
Define $S_l = 1, \ldots, l$. We have

$$
\sum_{i<j} \pi(i) M_{ij} (y(i)^2 - y(j)^2) = \sum_{i<j} \pi(i) M_{ij} \left( \sum_{i \leq l < j} (y(l)^2 - y(l+1)^2) \right)
$$

$$
= \sum_{l=1}^{k} (y(l)^2 - y(l+1)^2) \sum_{i \in S_l, j \notin S_l} \pi(i) M_{ij}
$$

$$
= \sum_{l=1}^{k} (y(l)^2 - y(l+1)^2) F(S_l).
$$

Since for $l \in [1, k, \pi(S_l) \leq \pi(S) \leq \frac{1}{2}$, we have

$$
F(S_l) \geq \psi(M) \pi(S_l).
$$

Therefore,

$$
\sum_{i<j} \pi(i) M_{ij} (y(i)^2 - y(j)^2) \geq \psi(M) \sum_{l=1}^{k} (y(l)^2 - y(l+1)^2) \pi(S_l)
$$

$$
= \psi(M) \sum_{l=1}^{k} (y(l)^2 - y(l+1)^2) \sum_{i=1}^{l} \pi(i)
$$

$$
= \psi(M) \sum_{i=1}^{k} \left( \sum_{l=1}^{k} (y(l)^2 - y(l+1)^2) \right) \pi(i)
$$

$$
= \psi(M) \sum_{i \in S} \pi(i) y(i)^2.
$$

Therefore,

$$
1 - \lambda \geq \frac{\psi(M)^2}{2}.
$$

(6.3)

(b) Proof of the leftmost inequality of equation (6.2). By Rayleigh’s theorem,

$$
1 - \lambda_2 \leq \frac{\varepsilon_\pi(x, x)}{\|x\|^2}
$$
for any vector nontrivial vector $x$ such that $\langle x \rangle_\pi = 0$. Select $B \subset E$ such that $\pi(B) \leq \frac{1}{2}$, and define

$$x(i) = \begin{cases} 1 - \pi(B) & \text{if } i \in B, \\ -\pi(B) & \text{if } i \notin B. \end{cases}$$

Then $\langle x \rangle_\pi = 0$ and $\|x\|_\pi^2 = \pi(B)(1 - \pi(B))$. Also,

$$\varepsilon_\pi(x, x) = \frac{1}{2} \sum_{ij} (x(i) - x(j))^2 \pi(i) M_{ij}$$

$$= \frac{1}{2} \left( \sum_{i \in B} \sum_{j \notin B} \pi(i) M_{ij} + \sum_{i \notin B} \sum_{j \in B} \pi(j) M_{ji} \right)$$

$$= \frac{1}{2} (F(B) + F(\bar{B}))$$

$$= F(B).$$

Therefore,

$$1 - \lambda_2 \leq \frac{F(B)}{\pi(B)(1 - \pi(B))} \leq 2 \frac{F(B)}{\pi(B)}.$$

So we have

$$\lambda_2 \geq 1 - 2\psi(M).$$

which concludes the proof.

\textbf{Proposition 3} Let $A = \lim_{n \to \infty} P^n$, then the conductance of $A$ is

$$\psi(A) \geq \frac{1}{2}.$$

\textbf{Proof.} We have

$$A = (\pi^T, \pi^T, \ldots, \pi^T)^T$$

136
by the Perron-Frobenius theorem. Thus

\[
\psi(B, A) = \frac{F(B)}{\pi(B)} = \frac{\sum_{i \in B, j \notin B} \pi(i)P_{ij}}{\pi(B)} = \frac{\sum_{i \in B, j \notin B} \pi(i)\pi(j)}{\pi(B)} = 1 - \pi(B)
\]

The conductance of the pair \((A, \pi)\)

\[
\psi(A) = \inf \left(1 - \pi(B); 0 < |B| < r, \pi(B) \leq \frac{1}{2} \right).
\]  
(6.4)

Apparently,

\[
\psi(A) \geq \frac{1}{2}.
\]

\(\square\)

Usually, for a large space on which \(\pi\) is distributed, we always could find a set \(B\) such that \(\pi(B) \approx \frac{1}{2}\). This gives us \(\psi(A) \approx \frac{1}{2}\) by equation (6.4). By equation (6.2) the SLEM of \(A\) is bounded as

\[
0 \leq \lambda_2 \leq 0.875.
\]

Since we know that the SLEM of \(A\) is, intuitively it converges at one step, 0. The upper bound derived here is quite loose. The important message from equation (6.2) though is the higher the conductance the smaller the SLEM. However, for many systems with big size of \(P\), it’s hard to achieve \(P\) with conductance as high as 0.5, and therefore, the obtained bounds are tight for small conductance.
Example 1 Let $\pi = (0.2 \ 0.06 \ 0.17 \ 0.04 \ 0.19 \ 0.07 \ 0.225 \ 0.045)^T$ be a target distribution and two transitions matrices,

$$P_1 = \begin{pmatrix}
0.33 & 0.05 & 0.10 & 0.00 & 0.21 & 0.00 & 0.31 & 0.00 \\
0.17 & 0.83 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
0.12 & 0.00 & 0.18 & 0.10 & 0.30 & 0.00 & 0.30 & 0.00 \\
0.00 & 0.00 & 0.43 & 0.58 & 0.00 & 0.00 & 0.00 & 0.00 \\
0.22 & 0.00 & 0.27 & 0.00 & 0.01 & 0.10 & 0.40 & 0.00 \\
0.00 & 0.00 & 0.00 & 0.00 & 0.27 & 0.73 & 0.00 & 0.00 \\
0.28 & 0.00 & 0.23 & 0.00 & 0.34 & 0.00 & 0.01 & 0.15 \\
0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.75 & 0.25 & 0.00
\end{pmatrix}$$

$$P_2 = \begin{pmatrix}
0.73 & 0.05 & 0.03 & 0.07 & 0.003 & 0.07 & 0.00 & 0.05 \\
0.17 & 0.61 & 0.00 & 0.22 & 0.00 & 0.00 & 0.00 & 0.00 \\
0.04 & 0.00 & 0.78 & 0.03 & 0.00 & 0.09 & 0.00 & 0.065 \\
0.35 & 0.33 & 0.13 & 0.09 & 0.00 & 0.10 & 0.00 & 0.00 \\
0.00 & 0.00 & 0.00 & 0.00 & 0.81 & 0.13 & 0.00 & 0.055 \\
0.20 & 0.00 & 0.22 & 0.06 & 0.35 & 0.17 & 0.00 & 0.00 \\
0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.96 & 0.045 & 0.00 \\
0.22 & 0.00 & 0.25 & 0.00 & 0.23 & 0.00 & 0.23 & 0.08
\end{pmatrix}$$

Denote $M_1 = P_1P_1$ and $M_2 = P_2P_2$, we have

$$\lambda_2(M_1) = 0.6986 \quad \text{and} \quad \lambda_2(M_2) = 0.9136.$$ 

By the definition of conductance,

$$\psi(M_1) = 0.2972 \quad \text{and} \quad \psi(M_2) = 0.0778.$$
From equation (6.2) the bounds can be computed as

\[ 0.4056 \leq \lambda_2(M_1) \leq 0.9558 \quad \text{and} \quad 0.8444 \leq \lambda_2(M_2) \leq 0.9970. \]

Apparently, the numbers computed are consistent and reflect the intuitive idea of connecting particles with big probability mass as to maximize the overall conductance. Thus the SLE is decreased and the convergence rate is improved.

[I found out that a bigger conductance not necessarily guarantees a larger SELM. Therefore, the theorem will be more concrete if we could prove a tighter upper bound.]

### 6.2 Designing Transition Matrix M With Big Conductance

This section includes some detailed design criterion for generating transition matrix with large conductance as for achieving fast convergence rate of Markov Chains. Since \( M = PP \) is just the second order transition matrix for the Markov Chain, we shall focus on \( M \) first and study its corresponding transition graph.

Given a target distribution \( \pi \), as in Dr. Zhu’s note, to meet the condition of \( \pi P = \pi \) and \( P \) being a stochastic irreducible and aperiodic transition matrix, there are, approximately,
2r constraints for \( r^2 \) unknowns. With \( r^2 - 2r \) freedom, we, almost always, could find a \( P \) that has a conductance close to 0.5 and a small SLEM near zero. However, such transition matrices ask for all to all connections to which we barely could afford since establishing each connection needs a certain amount of cost.

**Problem Definition** Let the cost of a transition matrix \( P \) defined as

\[
C(P) = \sum_{i \neq j} P_{ij}.
\]

What is the optimal transition matrix \( P \) of a reversible and irreducible Markov Chain that maximizes the conductance subject to \( C(P) < \text{cost} \)?

Instead of deriving the optimal \( P \) analytically, which is an NP-complete problem, we will propose several design criterion by which we could transform a “bad” \( P \) into a nearly optimal one.

**Proposition 4** Let \( B_1 \) and \( B_2 \) be two disconnected parts then

\[
\psi((B_1, B_2), P) \geq \min(\psi(B_1, P), \psi(B_2, P)).
\]

**Proof.** Because \( B_1 \) and \( B_2 \) are disconnected and by definition,

\[
\psi((B_1, B_2), P) = \frac{\pi(B_1)\psi(B_1, P) + \pi(B_2)\psi(B_2, P)}{\pi(B_1) + \pi(B_2)}.
\]

Without loss of generality, suppose \( \psi(B_1, P) \leq \psi(B_2, P) \), thus

\[
\psi((B_1, B_2), P) \geq \frac{\pi(B_1)\psi(B_1, P) + \pi(B_2)\psi(B_1, P)}{\pi(B_1) + \pi(B_2)} = \psi(B_1, P).
\]

\[ \square \]

**Proposition 4** tells us that the conductance doesn’t come from disconnected parts so that the burden of computing all possible combinations of particles could be eased to check connected parts only.

140
Criterion 1 For any particle with probability mass less than 0.5, it should not have small out flow.

\[
\psi(i, P) = \frac{F(i)}{\pi(i)} = \frac{\pi(i) \sum_{j \neq i} P_{ij}}{\pi(i)} = \sum_{j \neq i} P_{ij} = 1 - P_{ii}
\]  

Proof. By the definition, \( \psi(B, P) = \frac{F(B)}{\pi(B)} \), the ergodic flow out of each particle \( i \) is

Because \( \psi(P) < \psi(i, P) \) for any \( i \in E, \pi(i) \leq 0.5 \), a big value of \( P_{ii} \) will cause the conductance to be small. This criterion has the direct connection with the SLEM. For example, if \( P_{11} = 0.99 \), we will find out that at least one eigenvalue is very close to 0.99, which means the Markov Chain has very low convergence rate. It has a very intuitive interpretation that one eigenvector will be \( P_1 \) and 0.99 is the corresponding eigenvalue since most the elements are zeros or nearly zero. Given an arbitrary \( P \), we should check it’s diagonals to make sure none of them is close to 1 or larger than one less the desired conductance. Otherwise, we should decrease its value even it means to increase the overall cost.

Criterion 2 Avoid the existence of articulation points, especially for those disconnect the graph into two big parts.
As shown in figure 6.2, the conductance of $B$ could be computed as

$$\psi(B, P) = \frac{\pi(i)P_{ij}}{\pi(B)},$$

which is a fairly small value due to the big probability mass of $B$ in comparison to that of $i$. Intuitively, the path between $i$ and $j$ now is a bottle neck which causes traffic jam around $i$ and $j$ and increases the convergence time.

**Criterion 3** Replacing connections through low mass particle with bridges over particles with high mass directory improves the transition matrix $P$. Graph 6.2 shows an example, where path between 2 and 3 is replaced with a connection joins 1 and 3 directly.

**Proof.** Considering the flow from $B = (1, 2, 3)$ for the original $P$, we have

$$\psi(B, P) = \frac{\pi(1)(1 - P_{11} - P_{12}) + \pi(2)(1 - P_{21} - P_{22} - P_{23}) + \pi(3)(1 - P_{32} - P_{33})}{\pi(1) + \pi(2) + \pi(3)}.$$
The flow for $P'$ after the move is
\[
\psi(B, P') = \frac{\pi(1)(1 - P'_{11} - P'_{13}) + \pi(2)(1 - P'_{21} - P'_{22}) + \pi(3)(1 - P'_{31} - P'_{33})}{\pi(1) + \pi(2) + \pi(3)}.
\]

Let $P_{32} = P'_{31}$ and keep the flow out of each particle the same so that $\psi(B, P) = \psi(B, P')$, then the cost could be computed as
\[
C(P) - C(P') &= P'_{11} + P'_{22} + P'_{33} - (P_{11} + P_{22} + P_{33}) \\
&= P_{11} - P'_{13} + P_{22} + P_{23} + P_{33} - (P_{11} + P_{22} + P_{33}) \\
&= P_{23} - P'_{13} = P_{23} - \frac{P_{31}\pi(3)}{\pi(1)} = P_{23} - \frac{\pi(3)P_{32}}{\pi(1)} = P_{23} - \frac{\pi(2)P_{33}}{\pi(1)} \\
&= P_{23}(1 - \frac{\pi(2)}{\pi(1)}) > 0.
\]

If we compute all the $\psi$s that get changed due to the move of connection,
\[
F((1, 2), P') - F((1, 2), P) &= \pi(1)(1 - P'_{11} - P'_{12}) + \pi(2)(1 - P'_{21} - P'_{22}) \\
&\quad - \pi(1)(1 - P_{11} - P_{12}) - \pi(2)(1 - P_{21} - P_{22}) \\
&= \pi(1)(P_{11} - P'_{11}) + \pi(2)(P_{22} - P'_{22}) \\
&= \pi(1)P_{13} - \pi(2)P_{23} = \pi(2)P_{23} - \pi(2)P_{23} \\
&= 0,
\]
\[
F((2, 3), P') - F((2, 3), P) &= \pi(2)(1 - P'_{22}) + \pi(3)(1 - P'_{33}) \\
&\quad - \pi(3)(1 - P_{22} - P_{23}) - \pi(3)(1 - P_{32} - P_{33}) \\
&= \pi(3)P_{32} > 0,
\]
\[
F((1, 3), P') - F((1, 3), P) &= \pi(1)(1 - P'_{11} - P'_{13}) + \pi(3)(1 - P'_{31} - P'_{33}) \\
&= -\pi(3)P_{31} < 0.
\]
Apparently,

\[ F((1), P') - F((1), P) > 0, \quad F((3), P') - F((3), P) = 0, \]

\[ F((2), P') - F((2), P) < 0, \quad \text{and} \quad F((1, 2, 3, \ldots), P') - F((1, 2, 3, \ldots), P) = 0. \]

As long as we can keep \( \psi((1, 3), P') \) and \( \psi((2), P') \) above the current conductance, the overall cost is cut by \( P_{23} (1 - \frac{\pi(2)}{\pi(1)}) \).

**Criterion 4**

![Diagram](image)

Cutting one edge connecting two particles with small probability mass could, most likely, improve \( P \). Let \( P'_{22} = P_{24} + P_{22} \) and \( P'_{44} = P_{42} + P_{44} \), we observe

\[ \psi((1, 2), P') - \psi((1, 2), P) = -\frac{\pi(2)P_{24}}{\pi(1) + \pi(2)}. \]

If \( \pi(1) \gg \pi(2) \) then \( \psi((1, 2), P') - \psi((1, 2), P) \approx 0 \). The same rule applied on \( (3, 4) \). Thus, as long as we keep \( \psi((2), P') \), \( \psi((4), P') \) bigger than current conductance, the matrix \( P \) is improved in the sense that the overall cost decreases.

**Proposition 5** What’s left now is how to link the conductance of \( P \) and the conductance of \( M = PP \) together so to improve \( P \) is equivalent to improve \( M \) in conductance. For a
nonempty set \( B \subset E = 1, \ldots, r \), the ergodic flows out of \( B \) for \( P \) and \( M \) are respectively

\[
F(B) = \sum_{i \in B, j \notin B} \pi(i)P_{ij},
\]

and

\[
F'(B) = \sum_{i \in B, j \notin B} \pi(i)M_{ij}
\]

Since the two have the same capacity for any \( B \), we shall compare the ergodic flows only.

\[
F'(B) = \sum_{i \in B, j \notin B} \pi(i)M_{ij} = \sum_{i \in B} \pi(i) \sum_k P_{ik}P_{kj}
\]

\[
= \sum_k \pi(k) \sum_{i \in B} P_{ki} \sum_{j \notin B} P_{kj}
\]

\[
= \sum_{k \in B} \pi(k) \sum_{i \in B} P_{ki} \sum_{j \notin B} P_{kj} + \sum_{k \notin B} \pi(k) \sum_{i \in B} P_{ki} \sum_{j \notin B} P_{kj}
\]

\[
= \sum_{k \in B} \pi(k)(1 - \sum_{i \notin B} P_{ki})(\sum_{j \notin B} P_{kj}) + \sum_{k \notin B} \pi(k)(\sum_{i \in B} P_{ki})(1 - \sum_{j \notin B} P_{kj})
\]

\[
= F(B) + \sum_{k \notin B} \pi(k) \sum_{i \in B} P_{ki} \sum_{j \notin B} P_{kj} - \sum_{k \notin B} \pi(k) \sum_{i \in B} P_{ki} \sum_{j \notin B} P_{kj}
\]

\[
(6.7)
\]

It is, in general, not true for \((6.7)\) to be greater than \(F(B)\). Federic Guilloux found out that in the extreme case \(\psi(P)\) could be close to 1 while \(\psi(PP)\) is almost 0. Such a case can be seen as follows. Let \(\pi = (0.5, 0.25, 0.25)\) and \(P\) be

\[
P = \begin{pmatrix}
0.0002 & 0.4999 & 0.4999 \\
0.9998 & 0.0001 & 0.0001 \\
0.9998 & 0.0001 & 0.0001
\end{pmatrix}
\]

145
Thus,

\[
M = P \times P \begin{pmatrix} 0.9996 & 0.0002 & 0.0002 \\ 0.0004 & 0.4998 & 0.4998 \\ 0.0004 & 0.4998 & 0.4998 \end{pmatrix}.
\]

\(\psi(P) = 0.9998\) with eigenvalues \((1.0, 0.0, -0.9996)\) and \(\psi(M) = 0.0004\) with eigenvalues 
\((1.0, 0.9992, 0.0)\). However, if we add an additional constraints, \(\forall P_{ii} \geq x, i \in E\) to (6.7), then \(F'(B)\) is always greater than or equal to \(F(B)\).

\[
F'(B) = F(B) + \sum_{k \notin B} \pi(k) \sum_{i \in B} P_{ki} \sum_{j \notin B} P_{kj} - \sum_{k \notin B} \pi(k) \sum_{i \in B} P_{ki} \sum_{j \notin B} P_{kj}
\geq F(B) + x \sum_{k \notin B} \pi(k) - \sum_{k \in B} (1 - \sum_{i \in B} \pi(k) \sum_{i \in B} P_{ki})^2
\geq F(B) + xF(B) - (1 - x) \sum_{k \in B} \pi(k) \sum_{i \notin B} P_{ki}
= F(B) + (2x - 1)F(B)
\] (6.8)

We have proven that \(F'(B) \geq F(B)\) with \(x = 0.5\), which can easily to be used to show that \(\psi(M) \geq \psi(P)\). We reach a conclusion that to maximize \(\psi(M)\) is equivalent to maximize \(\psi(P)\) with \(\forall P_{ii} \geq 0.5, i \in E\).
CHAPTER 7

Future Work

The DDMCMC engine we devised in the thesis demonstrates its strength in combining both efficiency, by the taking advantages of using bottom-up techniques, and robustness, as from the use of Markov Chain Monte Carlo method under the Bayesian framework, and has shown promising results for a large set of natural images. There are many directions along which we could pursue to extend the system:

1. Recent progresses in face modeling [22, 23, 51, 91, 97] facilitate the modeling and computing of faces in natural images. We shall incorporate these new advances into the DDMCMC framework and explore how Ada Boost methods could be used to speed up the MCMC process.

2. This engine seems to well suit for extracting information for the descriptors in MPEG-7 standard [18, 56, 75, 81], which emphasizes on description of visual content. The stylish synthesized images in the previous chapters suggest some future applications of such an image parsing system for NPR and image compression.

3. We plan to study the properties of object such as the distributions of their shapes, positions and relations and incorporate them into the system.

4. To make the algorithm more theoretically sound, we will try to provide a rigorous analysis of DDMCMC convergence rate, which is not complete in chapter (6).
BIBLIOGRAPHY


