© Copyright by Kevin Michael Nickels, 1996

TEXTURED IMAGE SEGMENTATION USING MARKOV RANDOM FIELDS : RETURNING MULTIPLE SOLUTIONS

BY

KEVIN MICHAEL NICKELS

B.S., Purdue University, 1993

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Electrical Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 1996

Urbana, Illinois

ABSTRACT

Traditionally, the goal of image segmentation is to produce a single partition of an image. This partition is then compared to some "ground truth," or human approved partition, to evaluate the performance of the algorithm. This thesis utilizes a framework for considering a range of possible partitions of the image to compute a distribution of possible partitions of the image. This is an important distinction from the traditional model of segmentation, and has many implications in the integration of segmentation and recognition research. The probabilistic framework that enables us to return a confidence measure on each result also allows us to discard from consideration entire classes of results due to their low cumulative probability. Several experimental results are presented using Markov random fields as texture models to generate distributions of segments and segmentations on textured images. Both simple, homogeneous images and natural scenes are presented.

Dedicated to my parents, Jack L. and Marian K. Nickels

ACKNOWLEDGMENTS

I would like to thank my parents, to whom this thesis is dedicated, for bringing me up with the inquisitive and adaptive mindset needed to do this research.

I would like to thank my advisor, Professor Seth Hutchinson, for providing a rich, creative environment in which to work. His ideas and guidance have been invaluable.

I would like to thank my fiancée, Betsy Furlong, for her loving and understanding while this work was going on. Between being a sounding-board for my ideas and a wonderful proofreader, she knows more about computer vision than most other biophysical chemists.

I would like to thank my peers in the Computer Vision and Robotics research group for their many helpful discussions. I am especially grateful to Steve LaValle and Becky Castaño, whose work in segmentation paved the way for this research.

TABLE OF CONTENTS

Chapter **Page**

1 INTRODUCTION 1

2 TEXTURE MODELING AND DISCRIMINATION 7
2.1 Introduction 7
2.1.1 Descriptive features 7
2.1.2 Generative features 8
2.2 Markov Random Fields 10
2.3 The Degradation Model 12

3 OBTAINING THE PROBABILITY OF HOMOGENEITY OF TWO REGIONS 14

4 SEGMENT CLASSES 18
4.1 Definitions 18
4.2 Uncovering the Probability Mapping on Classes of Segments 21

5 SEGMENTATION CLASSES 25
5.1 Definitions 25
5.2 Uncovering the Probability Mapping on Classes of Segmentations 26

6 ALGORITHM ISSUES 29
6.1 Computing the Most Probable Segments 29
6.2 Computing the Most Probable Segmentations 31
6.3 Using Beam Search 33

7 EXPERIMENTAL ANALYSIS AND ILLUSTRATIONS 35
7.1 Introduction 35
7.2 Texture Modeling 35
7.3 Segment Distributions 38
7.3.1 Brodatz textures 38
7.3.2 Unfamiliar texture mosaics 45
7.3.3 Natural scenes 50

7.4 Segmentation Distributions 56
7.4.1 Brodatz textures 56
7.4.2 Unfamiliar texture mosaics 64
7.4.3 Natural scenes 68

8 EXTENSIONS AND CONCLUSIONS 75
8.1 Extensions 75
8.2 Conclusion 76

REFERENCES 77

LIST OF TABLES

Table **Page**

4.1 All possible segments from Figure 4.1. 19
4.2 All possible segment classes from Figure 4.1 that contain R_1. 21

LIST OF FIGURES

Figure **Page**

1.1 Two visually distinct textures that cannot be discriminated using most of the popular texture models. . . . 3
1.2 Two visually distinct textures that cannot be discriminated using most of the popular texture models. . . . 3

2.1 The neighborhood structure for MRFs. . . . 11

4.1 A simple example for describing segments. . . . 19
4.2 The first four covers of $seg(\{R_1\}, \emptyset)$, (a) $seg(\{R_1\}, \emptyset)$. (a) is refined into (b), (b) into (c), and (c) into (d). . . . 23

5.1 Ground segmentation classes for our simple example in decreasing order of probability. . . . 27
5.2 Segmentation classes for our simple example. Ground segmentations are in bold. . . . 28

6.1 An algorithm returning the n most probable segments. . . . 30
6.2 An algorithm returning the n most probable segmentations. . . . 32
6.3 An algorithm that performs beam search on the space of segmentations. . . 34

7.1 A 256x256 texture mosaic with much high-frequency content. . . . 36
7.2 A 256x256 texture mosaic with little high-frequency content. . . . 36
7.3 An image of wallpaper. . . . 37
7.4 Test data for Section 7.3.1. . . . 39
7.5 The 20 top segments. The region size is 64. There are 23 models. Image filename: brod1-64-23. . . . 40
7.6 The 20 top segments. The region size is 64. There are 23 models. Image filename: brod7-64-23. . . . 41
7.7 The 20 top segments. The region size is 64. There are 23 models. Image filename: brod9-64-23. . . . 42
7.8 The 20 top segments. The region size is 64. There are 23 models. Image filename: brod12-64-23. . . . 43

7.9 The 20 top segments. The region size is 64. There are 23 models. Image filename: brod13-64-23. 44
7.10 Test data for Subsection 7.3.2. 46
7.11 The 20 top segments. The region size is 64. There are 23 models. Image filename: qpatch2-64-23. 47
7.12 The 20 top segments. The region size is 64. There are 23 models. Image filename: F1-64-23. 48
7.13 The 20 top segments. The region size is 64. There are 23 models. Image filename: F3-64-23. 49
7.14 Test data for Subsection 7.3.3. 51
7.15 The 20 top segments. The region size is 64. There are 23 models. Image filename: F5-64-23. The initial region was on the top row, first column (Region 0). 52
7.16 The 20 top segments. The region size is 64. There are 23 models. Image filename: F5-64-23. The initial region was on the third row, eighth column (Region 58). 53
7.17 The 20 top segments. The region size is 64. There are 23 models. Image filename: F5-64-23. The initial region was on the eighth row, eighth column (Region 63). 54
7.18 The 20 top segments. The region size is 64. There are 23 models. Image filename: F7-64-23. 55
7.19 Test data for Subsection 7.4.1. 57
7.20 The 20 top segmentations. The region size is 64. There are 23 models. Image filename: brod1-64-23. 58
7.21 The 10 top segmentations. The region size is 128. There are 23 models. Image filename: brod4-128-23. 59
7.22 The 20 top segmentations. The region size is 64. There are 23 models. Image filename: brod7-64-23. 60
7.23 The 20 top segmentations. The region size is 64. There are 94 models. Image filename: brod9-64-94. 61
7.24 The 20 top segmentations. The region size is 64. There are 23 models. Image filename: brod12-64-23. 62
7.25 The 20 top segmentations. The region size is 64. There are 23 models. Image filename: brod13-64-23. 63
7.26 Test data for Subsection 7.4.2. 64
7.27 The 20 top segmentations. The region size is 64. There are 23 models. Image filename: qpatch2-64-23. 65
7.28 The 20 top segmentations. The region size is 64. There are 23 models. Image filename: F1-64-23. 66
7.29 The 20 top segmentations. The region size is 64. There are 23 models. Image filename: F3-64-23. 67
7.30 Test data for Section 7.4.3. 69

7.31 The 20 top segmentations. The region size is 128. There are 94 models. Image filename: F4-128-94. 70
7.32 The 20 top segmentations. The region size is 64. There are 23 models. Image filename: F5-64-23. 71
7.33 The 20 top segmentations. The region size is 128. There are 23 models. Image filename: F6-128-23. 72
7.34 The 20 top segmentations. The region size is 64. There are 23 models. Image filename: F7-64-23. 73
7.35 The 20 top segmentations. The region size is 64. There are 23 models. Image filename: F8-64-23. 74

CHAPTER 1

INTRODUCTION

A common task in computer vision is to partition an image into regions that are both maximal and homogeneous in some sense. This task, called *segmentation*, can also be defined as the process that subdivides a sensed scene into its constituent parts or objects [30].

Segmentation of an image is a complex but extremely important task in computer vision. Segmentation is usually conceded to be a necessary step before high-level vision algorithms such as object description, recognition, or scene analysis.

In addition to its uses as a "preprocessing step" in other areas of computer vision, segmentation itself is an important tool. Automatically finding regions of a medical image that differ from the rest of the image (possible tumors, for example), identifying different types of vegetation in multispectral images [2], and differentiating objects from backgrounds in frequencies not visible to the human eye are only a few of the tasks that computer image segmentation could accomplish.

To segment an image, one or more features must be extracted from the image. The term *feature* is often used in computer vision to refer to geometric or structural objects such as edges or corners. In statistical pattern recognition, a *feature* can be any observable characteristic of a picture, usually one that varies in an interesting way over the image.

We will take the latter definition, and refer to features such as color or texture. A *feature vector* is a vector whose entries are features.

Intensity and color are commonly used features. Consider an image of a dark asphalt road with bright green grass on either side. An algorithm would have no problem picking out the road using only color. On the other hand, an algorithm using only color would not be as effective at looking for a dull steel pipe under seven feet of murky water.

It is common to specify parameterized mathematical models that approximate parts of the image. In this case, the parameters of the model serve as features. Consider a range image of a toy block. By modeling the scene as consisting of piecewise planar surfaces, good segmentation results can be achieved. For this case, features might include the surface normal of the planar surfaces. It is easy to find cases where this feature vector would fail, however. For example, using this model on a range image of the moon's surface, where nothing is very regular, will not likely yield good segmentation results.

Texture is another popular feature to use when segmenting images. Texture in this context means a pattern of grey scale variation in an intensity image. Figure 1.1 shows examples of two visually distinct *textures*. The text in a newspaper can be considered to have a different texture from the greyscale pictures or line drawings. This fact can be used to allow the computer to read the text in a newspaper and to ignore the images when doing optical character recognition [39]. Textiles can be inspected for defects by searching for the inconsistencies in texture caused by a manufacturing problem [17]. Mathematical models for texture are not yet very well developed. For example, many texture models in use today cannot discriminate between some synthetic textures that are visually distinct, such as those in Figures 1.1 and 1.2 [21]. Many texture models have difficulty distinguishing between naturally occurring textures as well.

Most segmentation algorithms make use of one of two main approaches in dealing with features: they look for discontinuities in features or they look for similarities in

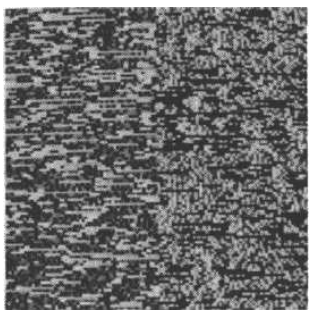


Figure 1.1 Two visually distinct textures that cannot be discriminated using most of the popular texture models.

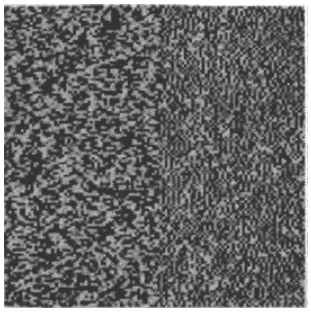


Figure 1.2 Two visually distinct textures that cannot be discriminated using most of the popular texture models.

features. Algorithms that look for discontinuities in features representing the borders of regions are called edge-based methods [27], [43]. Horn gives a review of edge-based segmentation methods [37]. Algorithms that look for similarities in features within two regions to decide whether the regions are homogeneous are classified as region-based methods. Some attempts have been made at developing algorithms that integrate these methods [15], [34]. Du Buf, Kardan, and Spann give a review of the performance of many types of features in image segmentation [24].

One region-based method is the split-and-merge procedure [57], which splits an image into small regions, then iteratively joins regions that are similar. This procedure can terminate based on the number of remaining regions, or on some other termination criterion [5], [54], [63], [64].

Another class of region-based methods is known as region growing algorithms. In region growing, the method begins with a few pixels or a small region, then adds neighboring pixels or groups of pixels to the region under consideration [59]. There are many variations on this, with region growing also being used to augment other methods [51].

In most region-based techniques, two regions will be joined if their feature vectors are similar. The makeup of these vectors differentiates between these approaches. Distance (in range images) or intensity value (in greyscale images) is a simple but naïve feature to use in segmentation. Variations on the use of this feature lead to a number of thresholding approaches to segmentation [58]. There are well-known problems with global thresholding [65]. An approach has been proposed that uses statistics from localized histograms (histograms using information from only a local neighborhood) to avoid some of these problems [5].

The facet model uses estimated parameters of planar, quadric, or spline surfaces as feature vectors [4], [35], [51], [66]. This means that patches of the image are modeled as planes or more complex surfaces, and parameters from these surfaces are used to guide the algorithms. This idea has been extended to the more complex surfaces described by explicit and implicit polynomial models as well [48].

Often in textured images, the texture in the image is modeled as an instance of a random field. In this case, parameters of the random field that is assumed to have generated the texture are used as features. Markov random fields are popular in this context since they model the local dependencies that exist between neighboring pixels in textured images [12], [20], [22], [25], [29], [40], [54]. Gibbs random fields and autoregressive random fields are also used.

Another approach to segmentation is to perform operations on multiple resolutions of an image, as opposed to performing all operations at the single resolution of the input

image. This type of algorithm is often called multiresolution or hierarchical segmentation [1], [7], [19], [52], [55], [60].

Most segmentation algorithms today restrict the set of possible solutions using various constraints to obtain a unique stable solution to the problem [41]. LaValle and Hutchinson [49] propose a region-based method that modifies the split-and-merge paradigm to create a set of possible solutions, and returns this set. This framework has been extended to identifying symmetries by Castaño [10]. This thesis extends the probabilistic framework proposed by LaValle to textured images, and refines much of the notation proposed by LaValle and Castaño.

This thesis is organized as follows. In Chapter 2 we discuss texture modeling and discrimination. First, we discuss the difference between descriptive and generative features, and give examples of each. Then, we expand the concept of generative features to arrive at the idea of texture models and present the Markov random field texture model, which will be used for the remainder of the thesis. Finally, we describe how we model the difference between our predictions and the input textures. Given the model for texture, in Chapter 3 we derive the probability that two disjoint regions are homogeneous. Then in Chapter 4 we combine the texture model from Chapter 2 and the probability of homogeneity expression from Chapter 3, with the result being a distribution of possible segments in the image. Each segment in this distribution has a confidence measure associated with it. Precise algorithms are presented later, in Chapter 6. The framework for segments developed in Chapter 4 is extended in Chapter 5 to consider segmentations of an image. Again, only the general framework is initially presented, with the actual procedures being detailed in the following chapter. Three algorithms are presented in Chapter 6. First, an algorithm for finding the distribution of segments presented in Chapter 4 is presented. Second, an algorithm for finding the distribution of segmentations from Chapter 5 is presented, along with a more efficient version that is based on

beam search. In Chapter 7, experimental results showing distributions of segments and segmentations, with the associated confidence measure from several different image sets, are presented and analyzed. Finally, possible extensions and conclusions to this research are presented.

CHAPTER 2

TEXTURE MODELING AND DISCRIMINATION

2.1 Introduction

Some goals in computer vision require a model for a mathematical process that could create the intensity values for a texture. Often, however, it is sufficient to extract some feature from a texture that distinguishes it from other textures in the image. Chen and Dubes [13] make this point by defining descriptive and generative textural features. A descriptive feature, such as roughness or directionality, can describe a texture, but could not be used to generate one. Generative features are sufficient to produce the intensity values that comprise a texture, as well as to describe it. The distinction between generative and descriptive features is similar to the distinction between statistics and sufficient statistics. Statistics describe a process, but unless these are sufficient statistics, they can not re-create the process.

2.1.1 Descriptive features

One characteristic that differentiates between textures is spatial frequency content, or the variation in intensity in different directions. Many researchers [16], [27], [39], [59] use filters or filter banks to find discontinuities in texture as a first step in segmenting

images. Filtering can also be used to characterize the frequency content at different orientations (i.e., amount of variation of intensity value in the vertical direction and amount of variation in the horizontal direction) [39], [59]. Other tools, such as transforms [64], Wold decompositions [53], and frequency domain symmetries [6], have been used to investigate the frequency content of texture as well.

Many different features that do not directly exploit an image's frequency content have been used. Laws formulates a set of "texture energy measures" that are used to compute features [50]. Each mask is dedicated to finding a particular type of energy in the image, such as the local variance of a window. Co-occurrence matrices from the spatial gray-level dependence method [36] have been used in much the same fashion as Laws' energy measures to compute features. Most of these techniques attempt to capture structural characteristics of the texture, such as 'edge like', or 'spot like' qualities, in the spatial domain. This is commonly done by convolving some sort of specialized mask with the image, and either directly observing the output or observing changes in the output of these convolutions.

2.1.2 Generative features

Random fields are popular texture models that contain sufficient information to generate certain textures. A random field is a lattice of random variables such that any random variable is dependent on other random variables in the field [68]. As Daily notes, because random fields are stochastic processes, the pixels in an image may take on any of their allowed values, which means that all images can be generated [22].

Random fields are commonly used to generate textures for research. Causal random fields, a subclass in which each site depends only on sites coming before it in some order, are relatively simple to generate. In this case, some initial distribution is assumed for the

first site, with subsequent sites having a distribution conditioned on the value found for some set of previous sites [21]. A random number generator can be used to obtain the actual values. Non-causal random fields are in general much more difficult to generate, requiring iterative algorithms that can take hundreds of thousands of sweeps of the image to converge [33]. This is because each site is dependent on its neighbors. In this case, each site is usually changed by a small amount on each sweep, until the interaction between the sites is sufficiently close to the required interaction defined by the random field [18].

One popular type of random field is the Markov random field (MRF) [12], [20], [22], [25], [29], [40], [54]. The MRF approach models an image as a lattice of random variables, with each site, or point on the lattice, having an explicit dependence on all sites within a local neighborhood (which is less than the entire image), and no dependence on any sites outside this neighborhood. The MRF model is derived from the Ising model, developed to explain molecular interaction in ferromagnetic materials [38]. The general MRF model most often seen in computer vision was introduced by Geman and Geman [32]. Auto-regressive random fields are a subclass of MRFs [42], [44]. More details on MRFs will be given in Section 2.2. Gibbs random fields have been shown to be equivalent to Markov random fields, but are usually named differently to emphasize the use of Gibbs' energy formulation for the probability distribution [23], [28].

Occasionally more complex random fields are used to model the texture. A compound Gauss-Markov random field (GMRF), for instance, models a texture as a two-level process with a hidden GMRF and an observable set of Gaussian processes [55].

Other methods model local dependence, but do not use a random field as an underlying model. Some examples of this are the spatial gray-level dependence method, the gray-level run length method, and the gray-level difference method [21]. The spatial gray-level dependence method estimates second-order joint conditional probability density functions at various orientations and uses these to compute texture features. The

gray-level run length method computes texture features based on the length of gray-level runs at various orientations. A gray-level run is a set of linearly adjacent picture points having the same gray-level value. The gray-level difference method is very similar to the spatial gray-level dependence method, but allows for different intersample spacing, or for having the probability density functions defined on a subsampled portion of the image.

Gabor functions are a way of viewing frequency as a local phenomenon, rather than a global one. Normally, texture is thought of as a collection of pixels (spatial), or as the sum of sinusoids of infinite extent (spatial-frequency) [27]. Gabor observed that there can be intermediate representations, with frequency being viewed as a local phenomenon that varies with position throughout the image [31]. While Gabor worked in communications, this observation has led to texture segmentation schemes that utilize this joint space/spatial-frequency representation [27].

2.2 Markov Random Fields

An MRF formulation must model the dependency of a pixel's intensity value on the intensity values of its neighbors. For example, for a first-order MRF model, a pixel in a given texture is modeled as a linear combination of its four-neighbors.

More formally, let an image I be an RxC lattice whose sites are denoted by $\{(r,c)|0 \leq r < R, 0 \leq c < C\}$. Let $X_{(r,c)}$ be a random variable representing the intensity value of pixel (r,c) in the image. The range of each random variable is $\Lambda = \{0, 1, ..., L\}$ for some integer L. In eight-bit gray-scale intensity images, eight bits are reserved to hold the intensity value of each pixel, so $L = 2^8 - 1 = 255$. A discrete MRF is formally defined as any random field for which, on this lattice, for each pixel $I(r,c)$,

5	4	3	4	5
4	2	1	2	4
3	1	x	1	3
4	2	1	2	4
5	4	3	4	5

Figure 2.1 The neighborhood structure for MRFs.

$$P(X_{(r,c)} = x_{(r,c)} | X_{(r',c')} = x_{(r',c')}, (r',c') \neq (r,c))$$

$$= P(X_{(r,c)} = x_{(r,c)} | X_{(r',c')} = x_{(r',c')}, (r,c) \in N_{(r,c)}) \tag{2.1}$$

and

$$P(X_{(r,c)}) > 0 \tag{2.2}$$

where $N_{(r,c)}$ is the set of neighbors for pixel (r,c). Informally, this means that a pixel depends only on its neighbors in some finite neighborhood, which is typically less than the entire image.

The *order* of an MRF defines the size of $N_{(r,c)}$. Let x denote pixel (r,c). Let y denote the intensity value of pixel (r,c). For an MRF of order n, x depends on all neighbors marked with n or less in Figure 2.1. For example, consider a first order MRF. Pixel x interacts only with the pixels above, below, to the left, and to the right of x (those marked with a 1). Specifically, if we define u_1, u_2, u_3, and u_4 to be the relative weighting of the four-neighbors of x, the linear model defined by a first-order MRF model for the intensity value of (r,c) is

$$y - \mu_k = u_1(I(r-1,c) - \mu_k) + u_2(I(r+1,c) - \mu_k) + \tag{2.3}$$

$$u_3(I(r,c-1) - \mu_k) + u_4(I(r,c+1) - \mu_k), \tag{2.4}$$

where μ_k represents the mean over some region.

More generally, for an n^{th} order MRF, if we denote the intensity value for the pixel of the l^{th} parameter interaction (the l^{th} neighbor) by $T_l(x)$, (2.3) becomes

$$y - \mu_k = \sum_{l=1}^{N} u_l(T_l(x) - \mu_k). \tag{2.5}$$

This formulation was introduced by Kashyap and Chellappa in [42].

2.3 The Degradation Model

If a naturally occurring texture is modeled by a random field, the pixel values are unlikely to be those predicted by the random field model. This discrepancy must be modeled. One approach is to model the texture as an instance of the random field plus additive Gaussian noise and to discuss the *probability* that the two regions can be modeled by the same instance of the random field. This is the approach we take here.

We assume that a pixel x differs from the prediction given in (2.5) because of additive Gaussian noise. We use this assumption to derive a probability density for a given pixel's intensity value. This density allows us to relate the intensities for the pixels in some region of the image to our mathematical model for the texture in that region.

The probability density for the intensity of a pixel x is

$$p(y|\mathbf{u_k}) = (2\pi\sigma_k^2)^{-\frac{1}{2}} exp\left\{\frac{-1}{2\sigma_k^2}\left[x - \mu_k - \sum_{l=1}^{N} u_l(T_l(x) - \mu_k)\right]^2\right\} \tag{2.6}$$

where σ_k^2 is the variance of the intensity values over the region R_k.

The joint density over the points in a region R_k, while not a proper probability density function, has been shown to be a reasonable approximation in many previous experiments [19], [46], [47], [56], [66]. We denote the vector of points in a region R_k by $\mathbf{x_k}$, and the vector of intensity values associated with these points by $\mathbf{y_k}$. The degradation

model defines how the intensity values differ from those predicted by the noiseless texture model described in (2.5). Since we assume the noise to be an independent identically distributed process, this model is obtained by taking the product of the densities over each pixel in R_k:

$$p(\mathbf{y_k}|\mathbf{u_k}) = (2\pi\sigma_k^2)^{-\frac{|\mathbf{x_k}|}{2}} exp\left\{\frac{-1}{2\sigma_k^2}\sum_{x\in\mathbf{x_k}}\left[x-\mu_k-\sum_{l=1}^{N}u_l(T_l(x)-\mu_k)\right]^2\right\}. \tag{2.7}$$

The degradation model defines exactly how we assume the image differs from our model for textures. Recall that each R_k potentially has a different instance of the random field as a generating field.

CHAPTER 3

OBTAINING THE PROBABILITY OF HOMOGENEITY OF TWO REGIONS

Of particular interest in region-based algorithms is an expression for the probability that two disjoint, but usually adjacent, regions come from the same instance of a random field, or, equivalently, from instances of random fields with the same parameters. This probability is called the probability of homogeneity. Often, researchers will estimate parameters for a random field from the region data and use these estimated parameters to compute an approximation for this probability [45]. This leads to many problems relating to parameter estimation techniques [33].

We will instead follow LaValle and Hutchinson [49], taking the degradation model from Section 2.3 and using it to build a probability distribution over possible segments, where a *segment* is some region grouping in the image. The degradation model describes the difference between the predicted intensity values for a region and the observed intensity values. Next we will use the degradation model, denoted as $p(\mathbf{y_k}|\mathbf{u_k})$, to derive the probability that two regions are homogeneous.

We assume that each region has a *true* parameter value associated with it, which is not known. We denote this value by $\mathbf{u_k}$. Each region has some set of data associated with it, which we represent as a vector of intensity values, $\mathbf{y_k}$. We will make extensive use

of the probability density functions (pdfs) of $\mathbf{Y_k}$ and $\mathbf{U_k}$ (the random variables of which $\mathbf{y_k}$ and $\mathbf{u_k}$ are instances), which we denote $p(\mathbf{y_k})$ and $p(\mathbf{u_k})$, respectively. Let $p(\mathbf{y_k}, \mathbf{u_k})$ be the joint pdf. The joint pdf can be represented as

$$p(\mathbf{y_k}, \mathbf{u_k}) = p(\mathbf{y_k}|\mathbf{u_k})p(\mathbf{u_k}). \tag{3.1}$$

The pdf for $\mathbf{y_k}$ is given by the marginalizing integral

$$p(\mathbf{y_k}) = \int p(\mathbf{y_k}, \mathbf{u_k})d\mathbf{u_k} = \int p(\mathbf{y_k}|\mathbf{u_k})p(\mathbf{u_k})d\mathbf{u_k}. \tag{3.2}$$

Let $H(R_i \cup R_j) = true$ if and only if $R_i \cup R_j$ is homogeneous. To say that two regions are homogeneous is equivalent to asserting that the random fields that generated the textures have identical parameter values, or $H(R_i \cup R_j) = true$ implies $\mathbf{u_i} = \mathbf{u_k}$. For notational convenience, we will use H to represent the condition $H(R_i \cup R_j) = true$.

We now wish to derive $P(H|\mathbf{y_i}, \mathbf{y_j})$, the probability that $R_i \cup R_j$ is homogeneous given the data from both regions. We can apply Bayes' rule to obtain

$$\begin{aligned} P(H|\mathbf{y_i}, \mathbf{y_j}) &= \frac{p(\mathbf{y_i}, \mathbf{y_j}|H)P(H)}{p(\mathbf{y_i}, \mathbf{y_j})} \\ &= \frac{p(\mathbf{y_i}, \mathbf{y_j}|H)P(H)}{p(\mathbf{y_i}, \mathbf{y_j}|H)P(H) + p(\mathbf{y_i}, \mathbf{y_j}|\neg H)P(\neg H)}. \end{aligned} \tag{3.3}$$

The denominator of (3.3) is the normalizing factor from Bayes' rule over the binary sample space, $\{H, \neg H\}$. The prior probability of homogeneity $P(H)$ is the *a priori* probability that two regions should be merged, and is usually taken to be $1/2$ to denote no prior knowledge of the problem.

We can separate (3.3) into λ_r, those terms dealing with regions R_i and R_j, and λ_p, those dealing with priors. Specifically, if we define

$$\lambda_p = \frac{1 - P(H)}{P(H)}, \text{ and } \lambda_r(\mathbf{y_i}, \mathbf{y_j}) = \frac{p(\mathbf{y_i}, \mathbf{y_j}|\neg H)}{p(\mathbf{y_i}, \mathbf{y_j}|H)} \tag{3.4}$$

then we can write (3.3) as

$$P(H|\mathbf{y_i}, \mathbf{y_j}) = \frac{1}{1 + \lambda_p \lambda_r(\mathbf{y_i}, \mathbf{y_j})}. \tag{3.5}$$

We make the assumption that regions that are not homogeneous have parameter values that are independent. This is by no means the only assumption that could be made at this point. For example, if a facet model is used for the regions, and it is known that all surfaces are planes and that all planes are either parallel or perpendicular, this information could be modeled in $p(\mathbf{y_i}, \mathbf{y_j}|\neg H)$. Our assumption, formally stated as

$$p(\mathbf{y_i}, \mathbf{y_j}|\neg H) = p(\mathbf{y_i})p(\mathbf{y_j}), \tag{3.6}$$

allows us to write $\lambda_r(\mathbf{y_i}, \mathbf{y_j})$ as

$$\lambda_r(\mathbf{y_i}, \mathbf{y_j}) = \frac{p(\mathbf{y_i})p(\mathbf{y_j})}{p(\mathbf{y_i}, \mathbf{y_j}|H)}. \tag{3.7}$$

Noting that the condition H states that $\mathbf{u_i} = \mathbf{u_j}$, we can expand $p(\mathbf{y_i}, \mathbf{y_j}|H)$ as a marginal with respect to $\mathbf{u_{ij}} = \mathbf{u_i} = \mathbf{u_j}$:

$$p(\mathbf{y_i}, \mathbf{y_j}|H) = \int p(\mathbf{y_i}, \mathbf{y_j}|\mathbf{u_{ij}})p(\mathbf{u_{ij}})d\mathbf{u_{ij}}. \tag{3.8}$$

We next state that if two regions, R_i and R_j, share the same parameter value, $\mathbf{u_{ij}}$, then

$$p(\mathbf{y_i}, \mathbf{y_j}|\mathbf{u_{ij}}) = p(\mathbf{y_i}|\mathbf{u_{ij}})p(\mathbf{y_j}|\mathbf{u_{ij}}), \tag{3.9}$$

which states that if the parameter vector for these two regions, $\mathbf{u_{ij}}$, is known, the observed intensity values in the two regions are statistically independent [49]. Combining these results gives a *Bayes' factor*

$$\lambda_r(\mathbf{y_i}, \mathbf{y_j}) = \frac{[\int p(\mathbf{y_i}|\mathbf{u_i})p(\mathbf{u_i})d\mathbf{u_i}]\,[\int p(\mathbf{y_j}|\mathbf{u_j})p(\mathbf{u_j})d\mathbf{u_j}]}{\int p(\mathbf{y_i}|\mathbf{u_{ij}})p(\mathbf{y_j}|\mathbf{u_{ij}})p(\mathbf{u_{ij}})d\mathbf{u_{ij}}}. \tag{3.10}$$

Bayes' factors have been used to model selection between nested linear parametric models [67], to model selection between parametric image models [48], and for evidence evaluation in forensic science [3], [11].

In many cases, especially in classification, an algorithm is provided with a number of *design samples* – particular representatives of the patterns we want to classify [26]. If

these samples are labeled, the algorithm is said to be *supervised*. If the samples are not labeled, the algorithm is said to be *unsupervised*. Note that while statisticians consider a sample to be a set of class representatives, engineers tend to consider a sample to be a particular class representative. We shall follow the latter definition in this thesis.

An expansion of (3.10) in the supervised case, where u_c is the class representative for class c and there are C classes, simply replaces each integration with a summation over the class representatives

$$\lambda_r(\mathbf{y_i}, \mathbf{y_j}) = \frac{\left[\sum_{c=1}^{C} p(\mathbf{y_i}|\mathbf{u_c})p(\mathbf{u_c})\right]\left[\sum_{c=1}^{C} p(\mathbf{y_j}|\mathbf{u_c})p(\mathbf{u_c})\right]}{\sum_{c=1}^{C} p(\mathbf{y_i}, \mathbf{y_j}|\mathbf{u_c})p(\mathbf{u_c})}. \tag{3.11}$$

There is one class representative per class in (3.11). How this representative parameter vector is chosen has been of interest to statisticians for some time. One approach is to separate the training set into classes and find the maximum likelihood estimate of the parameter value u_c [26]. A simpler approach is taken in this thesis. We take one homogeneous sample from each class and compute the best estimate $\tilde{u}_c$ using standard singular value decomposition techniques [61]. The resultant set of $\tilde{u}_c$ is then used as the set of class representatives.

CHAPTER 4

SEGMENT CLASSES

In this chapter, we will introduce and illustrate the use of the concept of a *segment class*, which is simply a shorthand representation for groups of segments that share some characteristic. By considering segment classes carefully, we can discard entire classes of segments from consideration, much as is done in $\alpha - \beta$ pruning and several other well-known techniques in computer science.

4.1 Definitions

The input to the segmentation algorithm will be an *image*, I, of intensity values. The elements of I are assumed to be on a rectangular lattice, which leads to a standard adjacency definition. Note that the theory presented in Chapters 4 and 5, which was proposed in [46], does not depend on which adjacency relationship is used. This thesis uses a four-neighbor adjacency relationship, but other relationships could be used.

A *region*, R, is defined as some connected subset of I, where *connected* is in the usual sense of each picture element in the region being related to every other element through some series of four-neighbors. Two regions will be called *adjacent* if there exist some $I(r_i, c_i) \in R_i$ and $I(r_j, c_j) \in R_j$ that are four-neighbors of each other. A *segment* is

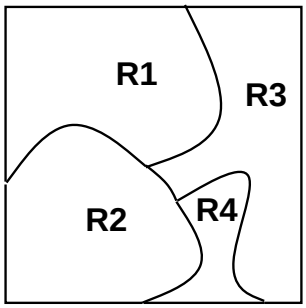


Figure 4.1 A simple example for describing segments.

Table 4.1 All possible segments from Figure 4.1.

seg_1	$\{R_1\}$
seg_2	$\{R_2\}$
seg_3	$\{R_3\}$
seg_4	$\{R_4\}$
seg_5	$\{R_1 \cup R_2\}$
seg_6	$\{R_1 \cup R_3\}$
seg_7	$\{R_2 \cup R_3\}$
seg_8	$\{R_2 \cup R_4\}$
seg_9	$\{R_3 \cup R_4\}$
seg_{10}	$\{R_1 \cup R_2 \cup R_3\}$
seg_{11}	$\{R_1 \cup R_2 \cup R_4\}$
seg_{12}	$\{R_1 \cup R_3 \cup R_4\}$
seg_{13}	$\{R_2 \cup R_3 \cup R_4\}$
seg_{14}	$\{R_1 \cup R_2 \cup R_3 \cup R_4\}$

defined to be a connected set of regions. We will denote a segment by seg. Consider a small example to illustrate these definitions. In Figure 4.1, there are four regions, R_1 through R_4. There are 14 possible segments shown in Table 4.1.

The ability to implicitly specify a set of segments that share some common characteristic will be very useful. In real image applications, the number of regions can be in the hundreds and the number of possible segments exponentially higher, so the advantage of an implicit representation is obvious. The number of possible groupings is exactly equal to the Bell number, or exponential number, in applications that do not require adjacency

in grouping [9]. Further, an implicit representation will allow us to calculate probabilities for a set of segments, instead of calculating them explicitly for each segment in the set.

We will call a set of segments a *segment class*. A segment class can be specified by two sets, an inclusion set, I, and an exclusion set, E. The *inclusion set* is a set of regions common to all segments in the segment class. The *exclusion set* is a set of regions contained in none of the segments in the segment class. In order to obtain a unique shorthand representation for each set of segments, we require each region in E to be adjacent to some region in I. A segment class may contain one or more than one segment. A segment class that contains only one segment shall be called a *singleton*, or *ground* class. We shall denote a segment class by $seg(I, E)$, where I and E are defined as above.

For example, the most general segment class for Figure 4.1 is $seg(\emptyset, \emptyset)$. A listing of the segments in this class is given in Table 4.2. This class represents all possible segments by indicating that the inclusion set is null, or that no particular set of segments must be included. Similarly, no particular set must be excluded. Therefore, every possible segment is included in $seg(\emptyset, \emptyset)$. An example of a class that contains only one segment is $seg(\{R_1\}, \{R_2, R_3\})$, which corresponds to the segment seg_1. Note that $R_4 \notin E$, since it is not adjacent to R_1. It can been shown that the triplet $(seg(\{R_i\}, \emptyset), PowerSet(seg(\{R_i\}, \emptyset)), P_i)$ is a probability space, where P_i is a probability mapping on $PowerSet(seg(\{R_i\}, \emptyset))$.

The next section describes how to select certain elements of $seg(\{R_i\}, \emptyset)$ in an organized manner, and how to apply evidence obtained by the models described in Section 2.2 to uncover P_i and find the most probable elements in $seg(\{R_i\}, \emptyset)$.

Table 4.2 All possible segment classes from Figure 4.1 that contain R_1.

Number	seg(I,E)	Contains segments
0	seg($\emptyset$,$\emptyset$)	seg_1,seg_2,seg_3,seg_4,seg_5,seg_6,seg_7,seg_8 seg_9,seg_{10},seg_{11},seg_{12},seg_{13},seg_{14}
1	seg($\{R_1\}$,$\emptyset$)	seg_1,seg_5,seg_6, seg_{10},seg_{11},seg_{12},seg_{14}
2	seg($\{R_1\}$,$\{R_2\}$)	seg_1,seg_6,seg_{12}
3	seg($\{R_1\}$,$\{R_3\}$)	seg_1,seg_5,seg_{11}
4	seg($\{R_1\}$,$\{R_2, R_3\}$)	seg_1
5	seg($\{R_1, R_2\}$,$\emptyset$)	seg_5,seg_{10},seg_{11},seg_{14}
6	seg($\{R_1, R_2\}$,$\{R_3\}$)	seg_5,seg_{11}
7	seg($\{R_1, R_2\}$,$\{R_4\}$)	seg_5,seg_{10}
8	seg($\{R_1, R_2\}$,$\{R_3, R_4\}$)	seg_5
9	seg($\{R_1, R_3\}$,$\emptyset$)	seg_6,seg_{10},seg_{12},seg_{14}
10	seg($\{R_1, R_3\}$,$\{R_2\}$)	seg_6,seg_{12}
11	seg($\{R_1, R_3\}$,$\{R_4\}$)	seg_6,seg_{10}
12	seg($\{R_1, R_3\}$,$\{R_2, R_4\}$)	seg_6
13	seg($\{R_1, R_2, R_3\}$,$\emptyset$)	seg_{10},seg_{14}
14	seg($\{R_1, R_2, R_3\}$,$\{R_4\}$)	seg_{10}
15	seg($\{R_1, R_2, R_4\}$,$\emptyset$)	seg_{11},seg_{14}
16	seg($\{R_1, R_2, R_4\}$,$\{R_3\}$)	seg_{11}
17	seg($\{R_1, R_3, R_4\}$,$\emptyset$)	seg_{12},seg_{14}
18	seg($\{R_1, R_3, R_4\}$,$\{R_2\}$)	seg_{12}
19	seg($\{R_1, R_2, R_3, R_4\}$,$\emptyset$)	seg_{14}

4.2 Uncovering the Probability Mapping on Classes of Segments

Consider R_j, a region adjacent to R_i. We can use $P(H_{ij})$ to describe the probability of the class of segments where R_i and R_j are homogeneous as

$$P\left[seg(\{R_i, R_j\}, \emptyset)\right] = P(H_{ij}) \tag{4.1}$$

where $P(H_{ij})$ is the probability that regions R_i and R_j are homogeneous. It has been shown [10] that a reasonable approximation to (4.1) using only information from regions

R_i and R_j is

$$P\left[seg(\{R_i, R_j\}, \emptyset)|I\right] \approx P\left[seg(\{R_i, R_j\}, \emptyset)|\mathbf{y_i}, \mathbf{y_j}\right] = P(H_{ij}|\mathbf{y_i}, \mathbf{y_j}) \tag{4.2}$$

which yields

$$P\left[seg(\{R_i\}, \{R_j\})|\mathbf{y_i}, \mathbf{y_j}\right] = P(\neg H_{ij}|\mathbf{y_i}, \mathbf{y_j}) = 1 - P(H_{ij}|\mathbf{y_i}, \mathbf{y_j}), \tag{4.3}$$

neatly splitting $seg(\{R_i\}, \emptyset)$ into two sets, the union of which makes up $seg(\{R_i\}, \emptyset)$. These two classes are called a *cover* of $seg(\{R_i\}, \emptyset)$, and this process of splitting a segment class is termed *refinement*.

For example, Figure 4.2 shows three refinements of $seg(\{R_1\}, \emptyset)$, under some $p(H_{12}|\mathbf{y_1}, \mathbf{y_2})$, into smaller classes. Consistent probabilities of homogeneity have been assigned in this example. Given the observations from R_1 and R_2, the probability that the two regions are homogeneous is 0.7. We split the segment class shown in (a) according to the homogeneity of R_1 and R_2 in the segments contained in the class. This yields the two classes shown in (b), the class containing all segments where R_1 and R_2 are homogeneous (the left part of (b)), or $seg(\{R_1, R_2\}, \emptyset)$, with probability $(1.0)(0.7) = 0.7$, and the class containing all segments where R_1 and R_2 are not homogeneous (the right part of (b)), or $seg(\{R_1\}, \{R_2\})$, with probability $(1.0)(1 - 0.7) = 0.3$. We then refine the highest probability segment class, which in this case is $seg(\{R_1, R_2\}, \emptyset)$ (the left part of (b)). Given the observations from R_{12} (the region formed by taking the union of R_1 and R_2) and R_3, the probability that the two regions are homogeneous is 0.14. The same grouping process described above is performed again. The segments that have R_{12} and R_3 homogeneous are grouped into $seg(\{R_1, R_2, R_3\}, \emptyset)$, shown on the left top of (c), which has probability $(0.7)(0.14) = 0.1$. Those segments that do not have R_{12} and R_3 homogeneous are grouped into $seg(\{R_1, R_2\}, \{R_3\})$, shown on the left bottom of (c), which has probability $(0.7)(1 - 0.14) = 0.6$. Similarly, $seg(\{R_1, R_2\}, \{R_3\})$ (left bottom of (c))

is refined into $seg(\{R_1, R_2\}, \{R_3, R_4\})$ (upper left bottom of (c)), which has probability $(0.6)(0.58) = 0.25$, and $seg(\{R_1, R_2, R_4\}, \{R_3\})$ (lower left bottom of (c)), which has probability $(0.6)(1 - 0.58) = 0.35$.

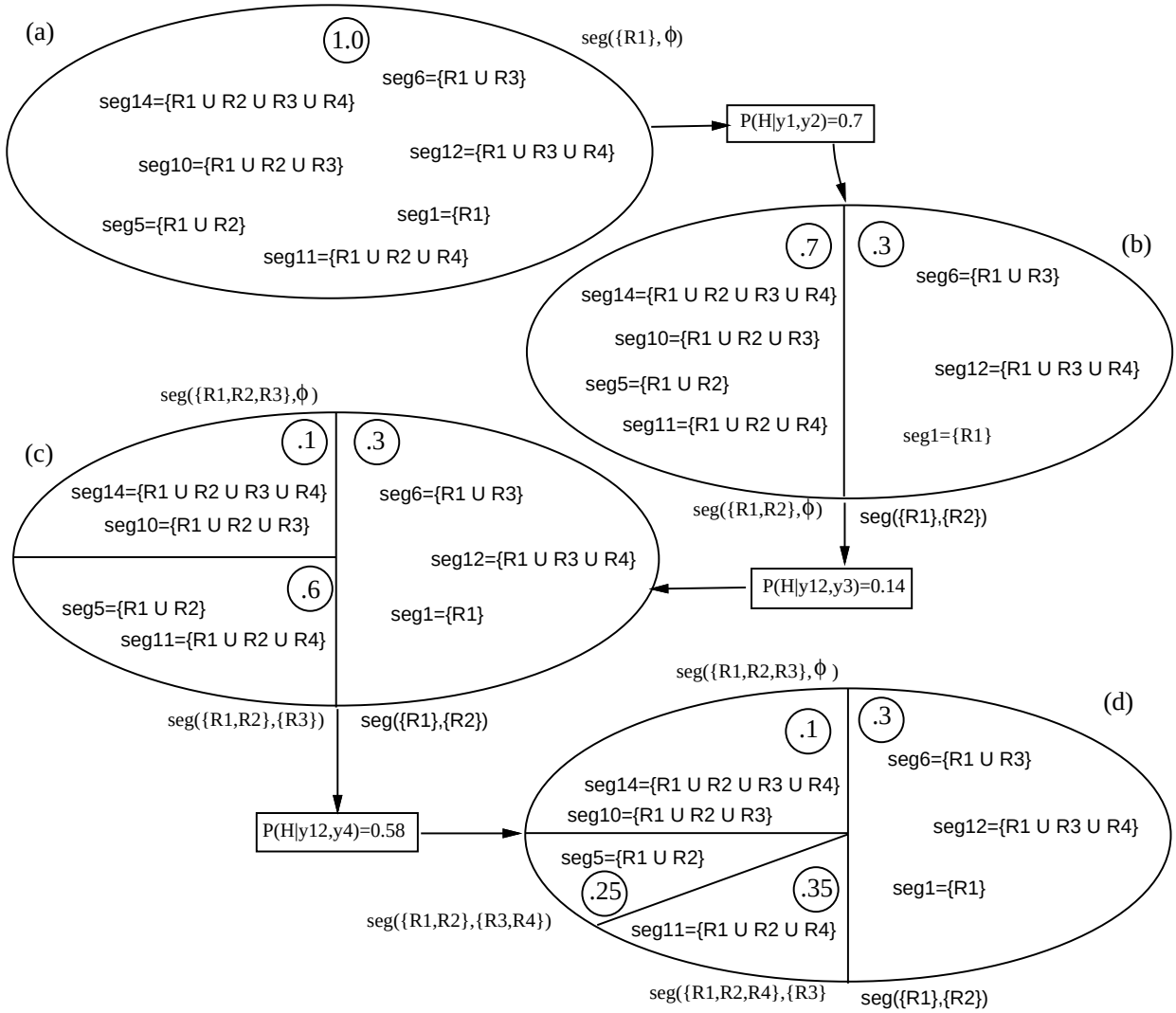


Figure 4.2 The first four covers of $seg(\{R_1\}, \emptyset)$, (a) $seg(\{R_1\}, \emptyset)$. (a) is refined into (b), (b) into (c), and (c) into (d).

This procedure takes advantage of the fact that the probability of a segment class is the sum of the probabilities of the segments that comprise it [10]. Also note that the probability of $seg(\{R_1, R_2, R_4\}, \{R_3\})$ is guaranteed to be higher than that of any segment in $seg(\{R_1\}, \{R_2\})$, because the probability of $seg(\{R_1\}, \{R_2\})$ itself is lower than that of $seg(\{R_1, R_2, R_4\}, \{R_3\})$. Therefore, if we are looking for the single most probable segment in $seg(\{R_1\}, \emptyset)$, we needn't look further at $seg(\{R_1\}, \{R_2\})$, even though it contains multiple segments. This computational savings can be very significant in large images, where huge classes of segments are not explicitly considered because of their low probability. In Section 6.1 we will formalize the process of refinement.

CHAPTER 5

SEGMENTATION CLASSES

In this chapter, we will introduce and illustrate the use of the concept of a *segmentation class*. A segmentation class, similar in flavor to the *segment class* of Chapter 4, is a compact representation for sets of segmentations.

5.1 Definitions

We define a *segmentation class* to be a set of possible partitions of the image I. For reasons that will become clear in Section 5.2, we choose to define a segmentation class in terms of some set of ground segment classes and at most one non-ground segment class. A *ground segmentation class* is defined to be a segmentation class that contains only ground segment classes. Just as a ground segment class contains exactly one segment, a ground segmentation class contains exactly one segmentation.

5.2 Uncovering the Probability Mapping on Classes of Segmentations

Like segment classes, segmentation classes have probabilities associated with them. The probability of a segmentation class is defined to be the product of the probabilities of the component segment classes. This implies certain independence relations, which we describe below.

The ground segments associated with a segmentation class specify a set of segments that are included in each segmentation of the segmentation class. For example, one segmentation class would be all those segmentations that contain $seg(\{R_1, R_2\}, \{R_3, R_4\})$. Like segment classes, segmentation classes that contain multiple elements can be refined to contain fewer elements.

The procedure for refining a segmentation class relies heavily on the procedure for refining a segment class. Specifically, we obtain a ground segment class from the procedure described in Chapter 4. Recall that a ground segment class contains a single segment. We remove this segment from the image. We then repeat the segment class refinement procedure to find a ground segment class from among the remaining regions. This is repeated until every region in the image has been included in some ground segment class.

Considering the segmentation classes for our simple example from Figure 4.1 will help to clarify this procedure. First, we use the procedure described in Chapter 4 to obtain all possible segments that contain R_1. For each of these segments, we consider the set of regions that remain after the segment is removed from the image. We refer to this set of regions as the residue. In the trivial case the residue is empty ((h) in Figure 5.1), and we have a ground segmentation class. If the residue is not empty, the procedure is used again to obtain all possible segments that make up the residue. If there is only one region left in the residue ((f),(h),(l) in Figure 5.1), that region will be returned as a segment.

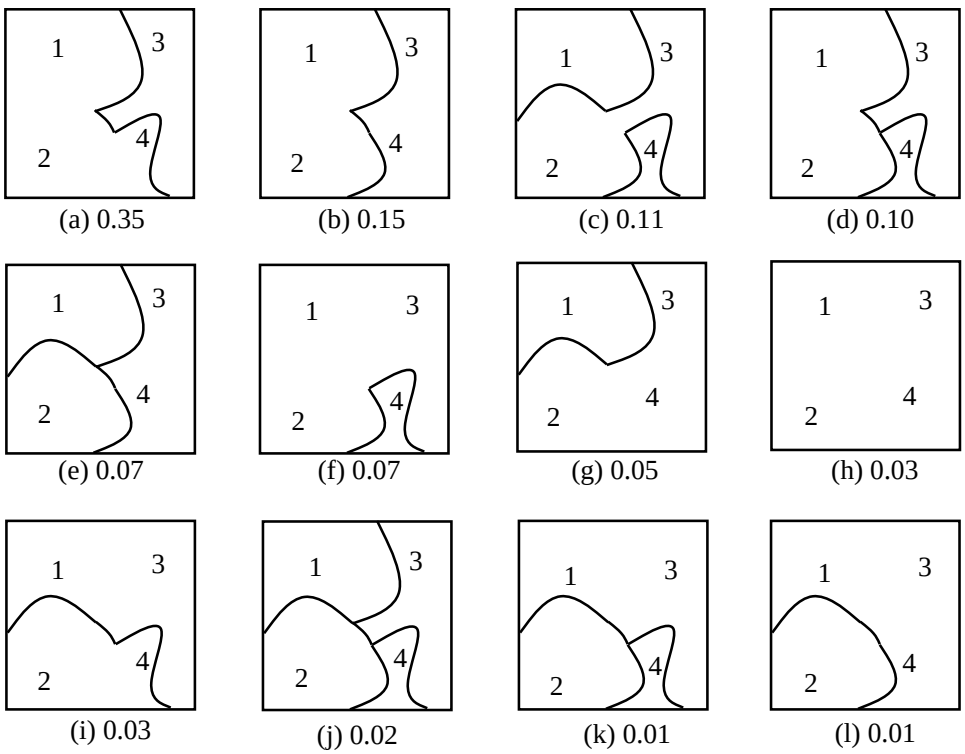


Figure 5.1 Ground segmentation classes for our simple example in decreasing order of probability.

Otherwise, multiple segments may be obtained from the residue. This process of finding all possible segments from a partial image continues until all regions in the image have been included in exactly one segment.

Recall that in Chapter 3 we made the assumption that regions with different parameter values are independent. Under this assumption, the probability mapping on a segmentation class is the product of the probabilities of the component classes. For example, consider the segmentation containing $seg(\{R_1, R_2\}, \{R_3, R_4\})$ and $seg(\{R_3\}, \{R_1, R_2\})$ in Figure 5.1. There is a 25% probability that the segment $seg(\{R_1, R_2\}, \{R_3, R_4\})$ is in the correct segmentation of the image I. Because this is a theoretical example, note that we can define a correct segmentation. In an actual image, it may not be clear which segmentation is correct. In the example, there is a 40% probability that the segment $seg(\{R_3\}, \{R_4\})$ is in the correct segmentation of the image $I - \{R_1 \cup R_2\}$. Therefore,

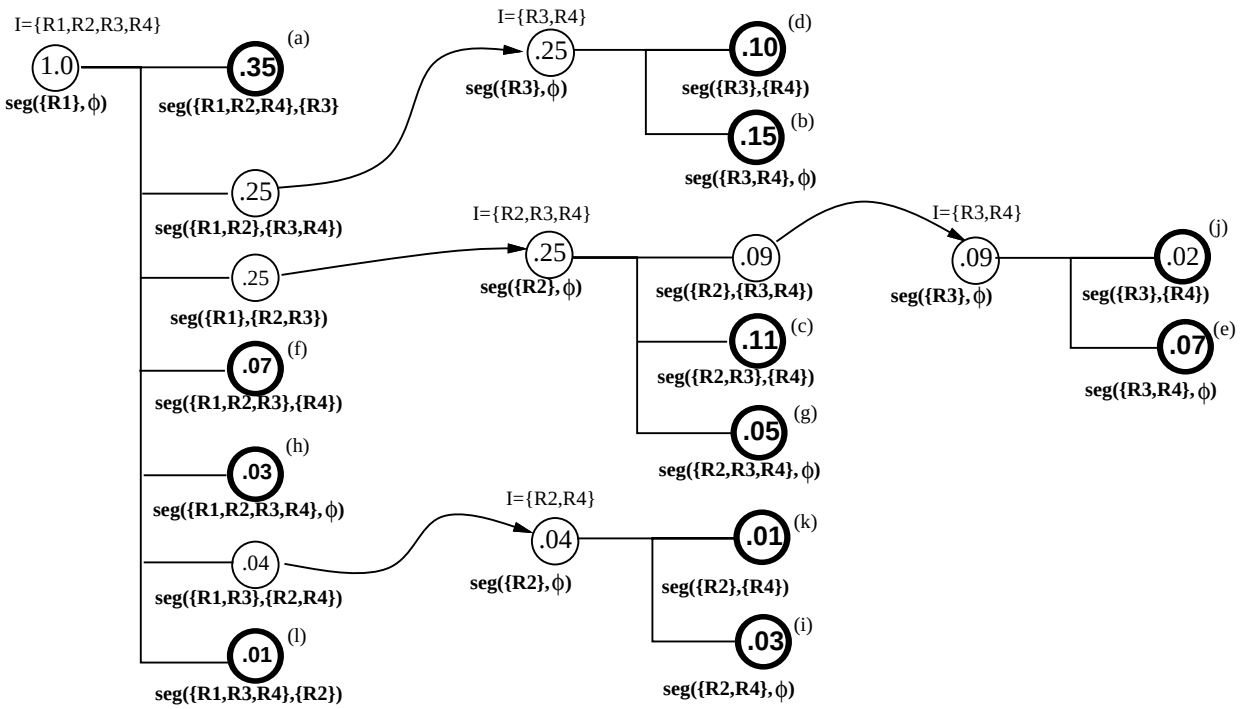


Figure 5.2 Segmentation classes for our simple example. Ground segmentations are in bold.

our definition of a $(0.25)(0.4) = 0.1$ probability that segments $seg(\{R_1, R_2\}, \{R_3, R_4\})$ and $seg(\{R3\}, \{R_1, R_2, R_4\})$ are in the correct segmentation of image I is reasonable. An illustration of this refinement process is shown in Figure 5.2. In the next chapter, we shall formalize procedures for obtaining the n most probable segmentations of an image.

CHAPTER 6

ALGORITHM ISSUES

In Chapters 4 and 5 we described procedures for obtaining the most probable segments and segmentations in an image. In this section, we will present algorithms for obtaining these. The first algorithm obtains the n segments with the highest probability from an image. The second algorithm obtains the n segmentations with the highest probability from an image. The third algorithm uses a beam search to obtain an approximation to the most probable n segmentations in an image.

6.1 Computing the Most Probable Segments

Given a list of regions in the image, $\mathcal{R}$, and an initial region, R_i, the algorithm shown in Figure 6.1 determines the n most probable segments and returns them. Two queues, $segQ$ and $segQ_g$, are maintained for the segment classes and ground segment classes that are generated, respectively. By setting $n = 1$, the most probable segment can be found. Since the highest probability segment class is selected to be refined at each iteration, $segQ$ and $segQ_g$ are implemented with priority queues to ensure the efficiency of this operation.

GET-MOST-PROBABLE-SEGMENTS $(\mathcal{R}, R_i, n)$

1 $segQ \leftarrow seg(\{R_i\}, \{\}); segQ_g \leftarrow \emptyset$; $k \leftarrow 0$
2 **repeat**
3 $k \leftarrow k + 1$
4 $seg_k(I_k, E_k) \leftarrow$ EXTRACTMAX(seg)
5 select R_k from $\mathcal{R} - I_k - E_k$
6 $seg_{k_I} \leftarrow seg(I_k \cup \{R_k\}, E_k)$
7 $seg_{k_E} \leftarrow seg(I_k, E_k \cup \{R_k\})$
8 calculate probabilities for seg_{k_I} and seg_{k_E}
9 **if** seg_{k_I} is a ground segment class
10 INSERT$(segQ_g, seg_{k_I})$
11 INSERT$(segQ_g, seg_{k_E})$
12 **else**
13 INSERT$(segQ, seg_{k_I})$
14 INSERT$(segQ, seg_{k_E})$
15 **until** $segQ = \emptyset$ OR
$(|segQ_g| \geq n$ AND $P(n^{th}\ element\ of\ segQ_g) > P(max\ element\ of\ segQ))$
16 **if** $(|segQ_g| > n)$ truncate $segQ_g$ to n elements
17 **return** $segQ_g$

Figure 6.1 An algorithm returning the n most probable segments.

Each iteration of the refinement loop in this algorithm splits the segment class under consideration into two segment classes, each of which is smaller than the original. Probabilities for the two new segment classes are calculated as discussed in Section 4.2.

It is possible that we will have to refine more than n segments in order to guarantee that there are no segments in segment classes from $segQ$ that could have higher probability than the n_{th} ground segment class in $segQ_g$. In this case, we truncate $segQ_g$ in step 15 before terminating.

6.2 Computing the Most Probable Segmentations

The algorithm that computes the n most probable segmentations, shown in Figure 6.2, utilizes a stepwise refinement process. The inputs to the algorithm are a list of regions in the image, $\mathcal{R}$, and an initial region, R_i. Two queues of segmentation classes, $\mathcal{SEGQ}$ and $\mathcal{SEGQ}_g$, are maintained. They contain non-ground segmentation classes and ground segmentation classes, respectively.

A segmentation class is denoted by $SEG\,[S, I, E]$, where S is a list of ground segment classes, and I and E are the inclusion set and the exclusion set in the one non-ground segment class in the segmentation class. See Section 4.1 for a review of these definitions. We denote by $\bar{S}$ the set of regions contained in any segment in S.

The only section of this algorithm that is not obviously part of the refinement described is the section given in lines 17 through 24. In these lines, we deal with the case in which the segment class under consideration is a ground segment class, but the segmentation class is not a partition of the entire image. Therefore, we have to find a new segment class to refine in the next step. If the next segment class we obtain happens to be a ground segment class, it cannot be refined, so we have to select a new segment class after adding the segment to S. If there are no remaining segment classes, we have refined SEG to a ground class, so we add the segmentation to $\mathcal{SEGQ}_g$, and begin again.

GET-MOST-PROBABLE-SEGMENTATIONS $(\mathcal{R}, n)$

1 $\mathcal{SEGQ} \leftarrow \emptyset; \mathcal{SEGQ}_g \leftarrow \emptyset$
2 select initial R_i
3 INSERT$(\mathcal{SEGQ}_g, SEG[\emptyset, \{R_i\}, \emptyset])$; $k \leftarrow 0$
4 **repeat**
5 $k \leftarrow k + 1$
6 $SEG[S_k, I_k, E_k] \leftarrow$ EXTRACTMAX$(\mathcal{SEGQ})$
7 select R_k from $\mathcal{R} - I_k - E_k$
8 $SEG_{k_I} \leftarrow SEG[S_k, I_k \cup \{R_k\}, E_k]$
9 $SEG_{k_E} \leftarrow SEG[S_k, I_k, E_k \cup \{R_k\}]$
10 calculate and store $P(SEG_{k_I})$ and $P(SEG_{k_E})$
11 **if** $seg(I_k \cup \{R_k\}, E_k)$ and $seg(I_k, E_k \cup \{R_k\})$ are ground classes
12 **if** $\mathcal{R} - I_k - R_k - \bar{S}_k = \emptyset$ [if this is a partition of I]
13 $SEG_{k_I} \leftarrow SEG[S_k \cup \{I_k \cup \{R_k\}\}, \emptyset, \emptyset]$
14 INSERT$(\mathcal{SEGQ}_g, SEG_{k_I})$
15 $SEG_{k_E} \leftarrow SEG[\{S_k \cup \{I_k\} \cup \{R_k\}\}, \emptyset, \emptyset]$
16 INSERT$(\mathcal{SEGQ}_g, SEG_{k_E})$
17 **else**
18 **repeat**
19 select R_{i_I} from $\mathcal{R} - I_k - R_k - \bar{S}_k$
20 **if** $seg(\{R_{i_I}\}, \emptyset)$ is a ground segment class
21 **if** $R_{i_I} = \emptyset$ [Partition]
22 INSERT$(\mathcal{SEGQ}_g, SEG[S_k, \emptyset, \emptyset])$
23 **continue** [to 4]
24 INSERT $(\mathcal{SEGQ}, S_k \leftarrow S_k \cup seg(R_{i_I}, \emptyset)$
25 **until** $seg(\{R_{i_I}\}, \emptyset)$ is not a ground segment class
26 INSERT$(\mathcal{SEGQ}, SEG[S_k \cup seg(I_k \cup \{R_k\}), \{R_{i_I}\}, \emptyset]$
27 **else**
28 INSERT$(\mathcal{SEGQ}, SEG_{k_I})$
29 INSERT$(\mathcal{SEGQ}, SEG_{k_E})$
30 **until** $\mathcal{SEGQ} = \emptyset$ OR
$(|\mathcal{SEGQ}_g| \geq n$ AND $P(n^{th}\ element\ of\ \mathcal{SEGQ}_g) > P(max\ element\ of\ \mathcal{SEGQ}))$
31 **return** $\mathcal{SEGQ}_g$

Figure 6.2 An algorithm returning the n most probable segmentations.

6.3 Using Beam Search

We have also implemented a beam search algorithm to compute an approximation to the n most probable segmentations. This algorithm is shown in Figure 6.3. In the interest of computational efficiency, BEAM-SEARCH-SEGMENTATIONS only considers the most probable segments at each step in the refinement process, whereas GET-MOST-PROBABLE-SEGMENTATIONS considers every possible segment. A result of this is that the segmentation classes are refined one segment at a time, rather than by regions as before.

The inputs to BEAM-SEARCH-SEGMENTATIONS are the list of all regions, $\mathcal{R}$, the number of segmentations to find, n, and the beam width, b. The refinement process is straightforward: a non-ground segmentation class is refined into b (or fewer) segmentation classes by adding the most probable b (or fewer) segments obtained from GET-MOST-PROBABLE-SEGMENTS. The termination criterion is analogous to that from GET-MOST-PROBABLE-SEGMENTS.

BEAM-SEARCH-SEGMENTATIONS $(\mathcal{R},\ n,\ b)$

```
1   SEGQ_g ← ∅; SEGQ ← ∅
2   select initial R_i
3   INSERT(SEGQ, SEG[∅, {R_i}, ∅])
4   repeat
5       SEG[S_k, ∅, ∅] ← EXTRACTMAX(SEGQ)
6       select new R_i from R − S̄_k
7       S_g ← GET-MOST-PROBABLE-SEGMENTS(R − S̄_k, R_i, b)
8       for seg(I_g, E_g) ∈ S_g do
9           if (R − S̄_k − I_g) = ∅ [Partition]
10              then
11                  INSERT(SEGQ_g, SEG[S_k ∪ I_g, ∅, ∅])
12              else
13                  INSERT(SEGQ, SEG[S_k ∪ I_g, ∅, ∅])
14  until SEGQ = ∅ OR
        (|SEGQ_g| ≥ n AND P(n^th element of SEGQ_g) > P(max element of SEGQ))
15  if (|SEGQ_g| > n) truncate SEGQ_g to n elements
16  return SEGQ_g
```

Figure 6.3 An algorithm that performs beam search on the space of segmentations.

CHAPTER 7

EXPERIMENTAL ANALYSIS AND ILLUSTRATIONS

7.1 Introduction

The algorithms presented in Chapter 6 were implemented using C++ under SunOS 5.4 (Solaris 2.4) and Linux 1.2. Experiments were conducted using image mosaics with subimages from the Brodatz album [8] and from MIT's Vision Texture database [70]. Distributions of segments and distributions of segmentations were generated for several test images.

7.2 Texture Modeling

It has been shown that Gauss Markov random fields do not fit many natural textures well, but are quite good at modeling many other textures [14]. In Chapter 3 we described how the probability of homogeneity computation follows from the MRF model, and in Chapter 4 we described how this probability contributes to the construction of a distribution of segments.

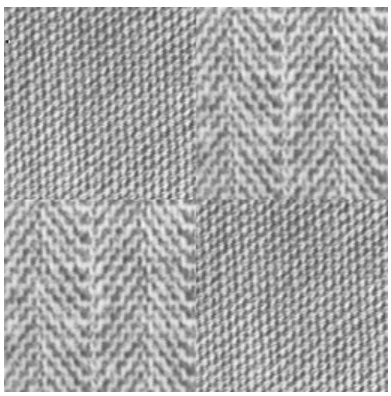


Figure 7.1 A 256x256 texture mosaic with much high-frequency content.

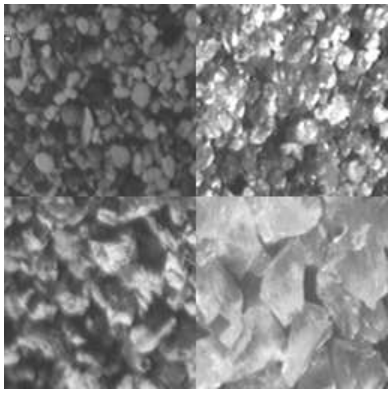


Figure 7.2 A 256x256 texture mosaic with little high-frequency content.

Since third-order MRFs have a small neighborhood, low-frequency intensity variations will not be detected using this model for texture discrimination. It has been noted that even on the homogeneous, uniformly lit images from the Brodatz album, small subimage sizes cause misclassification of some textures [62]. One consequence of this effect that was noted in our work is that a larger subimage size may be required for robust discrimination of textures without much high-frequency energy. For example, with the rapidly varying textures in Figure 7.1, a 256x256 mosaic of four 128x128 subimages, a subimage size of 64x64 pixels gave good results, but the textures shown in Figure 7.2 required a 128x128 subimage to discriminate between the different textures and thus give good results.

Figure 7.3 An image of wallpaper.

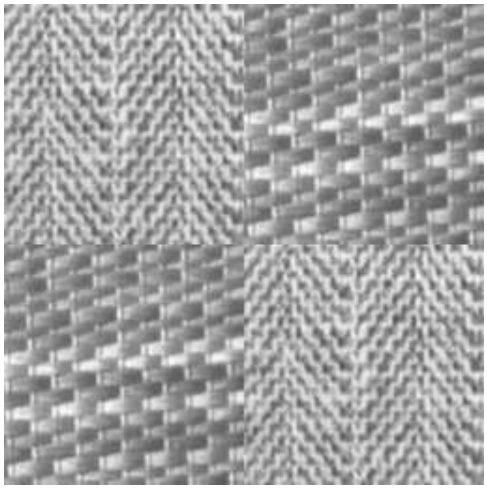
Two sets of class representatives were used in these experiments. Each class representative in the first set consisted of the estimated parameters of a third-order MRF fitted to one image from a subset of the Brodatz database. Specifically, we selected 24 images from the Brodatz database, each of which contained mostly homogeneous textures with significant high-frequency information. Each class representative consists of the estimated parameters of the MRF model based on one image from this set. Each class representative in the second set consisted of the estimated parameters of a third-order MRF fitted to one image from a much larger subset of the Brodatz database. Specifically, we selected 94 images from the database for this set. The parameters that were estimated are those given in (2.5). It is interesting to note that many of the textures in the Brodatz database do not contain the homogeneous high-frequency variations normally associated with the concept of texture. For example, consider Figure 7.3, which is a picture of wallpaper. Many of the images, like this one, either contain no regular repeating pattern or contain one too large for a small neighborhood structure to model. Thus the second set of class representatives contains several vectors that do not represent any information about the high-frequency variations of the type that a third-order MRF can detect. Possibly because of this, some textures presented below yield better segmentation results when the first set of class representatives is used.

7.3 Segment Distributions

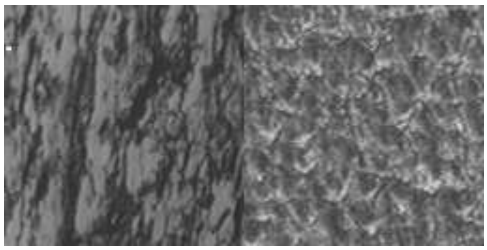
This section presents distributions of segments for some images. In each case, an initial region was chosen, and a distribution of segments was returned. The algorithm was tested on Brodatz textures, unfamiliar textures, and natural scenes.

7.3.1 Brodatz textures

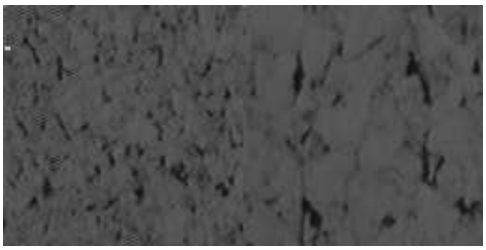
As described above, a selected group of textures from the Brodatz database were used in the generation of class representatives used in the experiments. The first category of experiments dealt with texture mosaics made from these textures, shown in Figure 7.4. This situation, predictably, gives the best results. Each result, shown in Figures 7.5, 7.6, 7.7, 7.8, and 7.9 is labeled with the size of the set of class representatives used in the experiment. For example, the first set was used in Figure 7.5 so the caption reads "There are 23 models". Region 0 (the region in the top left corner) was used as the initial region in all experiments where the initial region is not labeled in the caption.



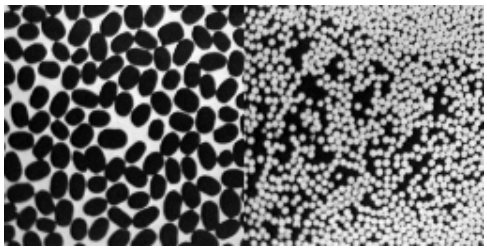
brod1



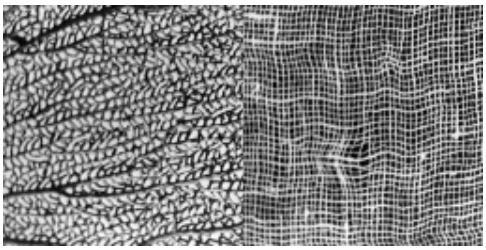
brod7



brod9

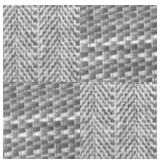


brod12

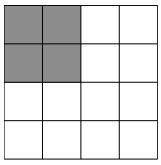


brod13

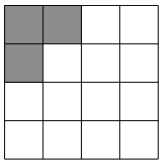
Figure 7.4 Test data for Section 7.3.1.



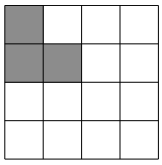
Input Image



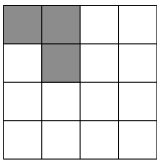
1: 9.99869e-1



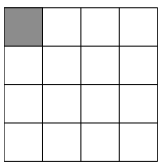
2: 4.35161e-5



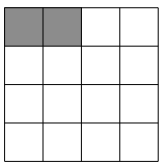
3: 4.35161e-5



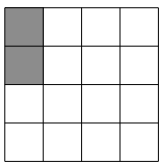
4: 4.35161e-5



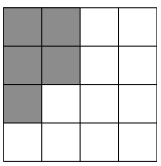
5: 1.89398e-9



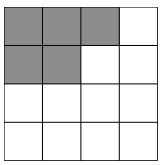
6: 1.8939e-9



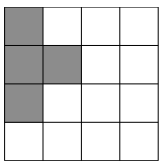
7: 1.8939e-9



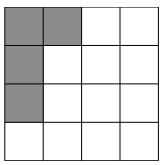
8: 1.36408e-16



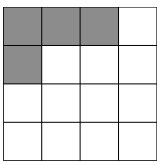
9: 1.7969e-17



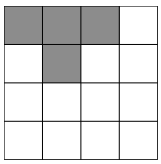
10: 5.93672e-21



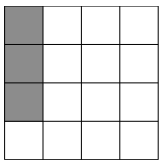
11: 5.93672e-21



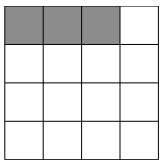
12: 7.82045e-22



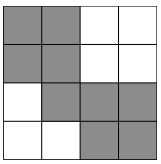
13: 7.82045e-22



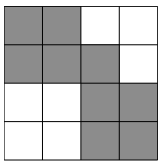
14: 2.58377e-25



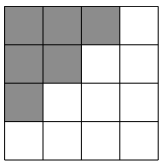
15: 3.4036e-26



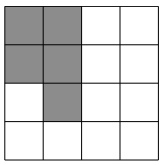
16: 3.43179e-31



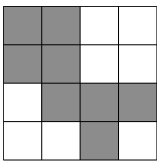
17: 8.24208e-33



18: 2.45144e-33

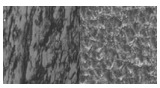


19: 1.49377e-35



20: 1.49358e-35

Figure 7.5 The 20 top segments. The region size is 64. There are 23 models. Image filename: brod1-64-23.



Input Image

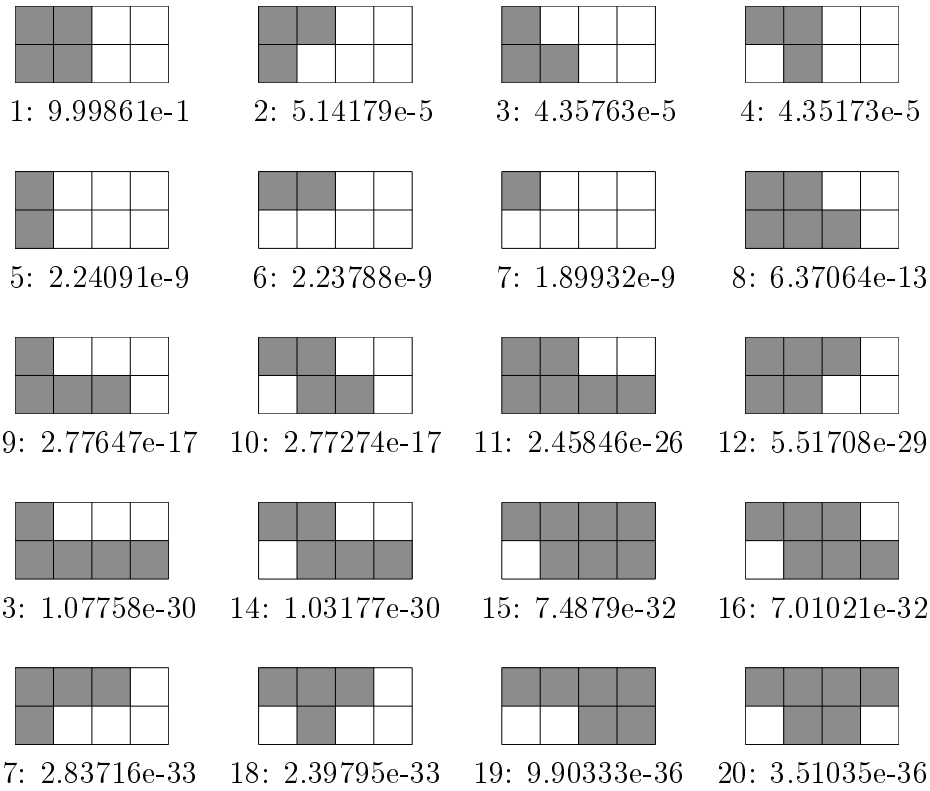


Figure 7.6 The 20 top segments. The region size is 64. There are 23 models. Image filename: brod7-64-23.

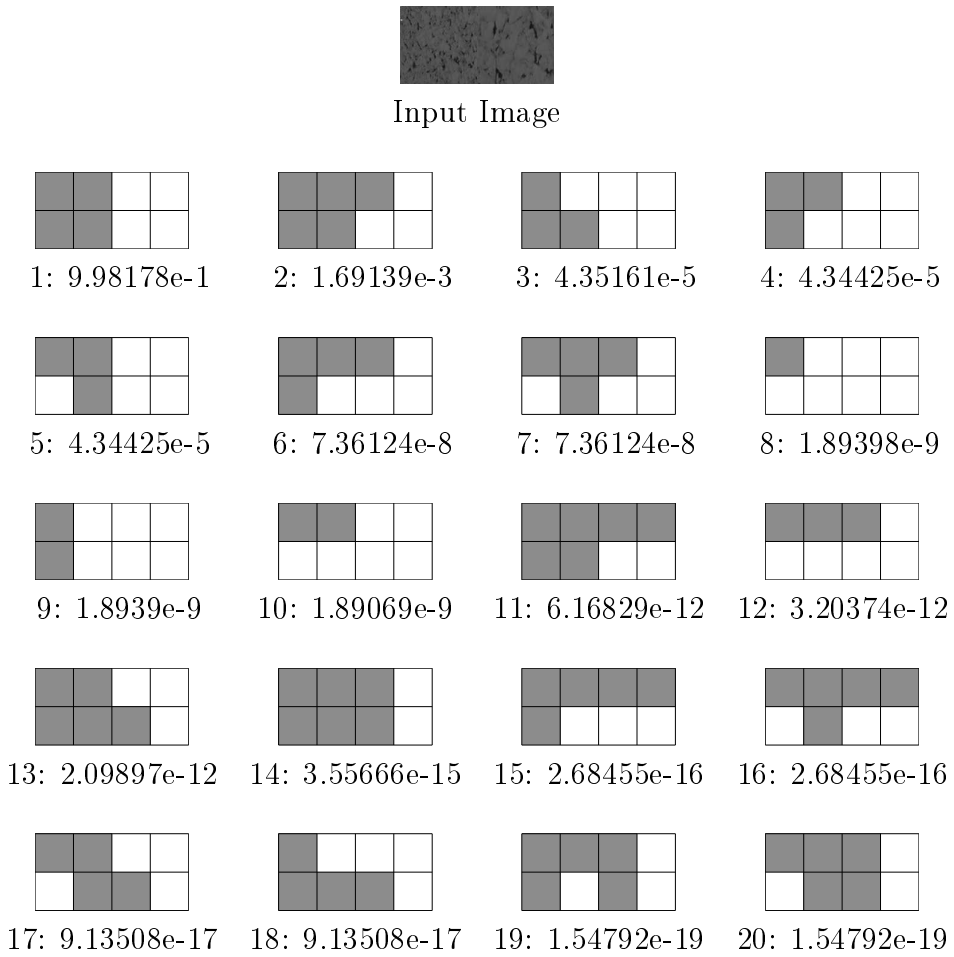


Figure 7.7 The 20 top segments. The region size is 64. There are 23 models. Image filename: brod9-64-23.

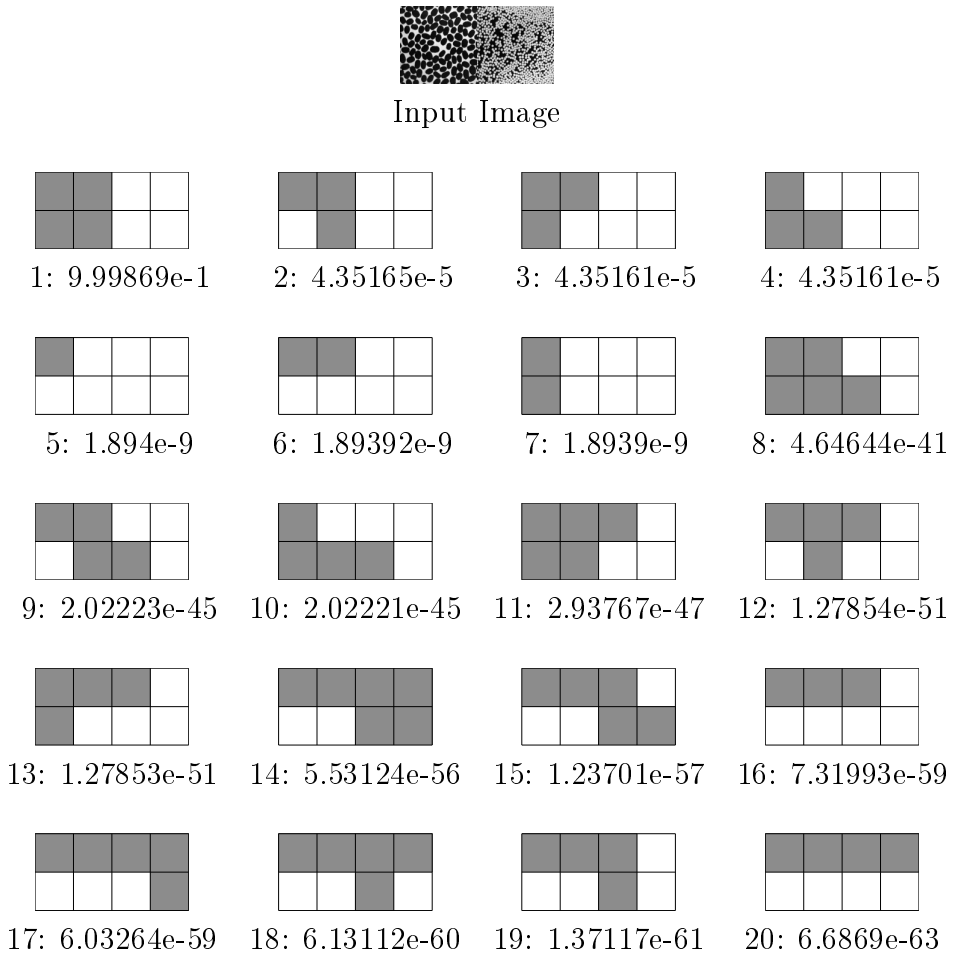


Figure 7.8 The 20 top segments. The region size is 64. There are 23 models. Image filename: brod12-64-23.

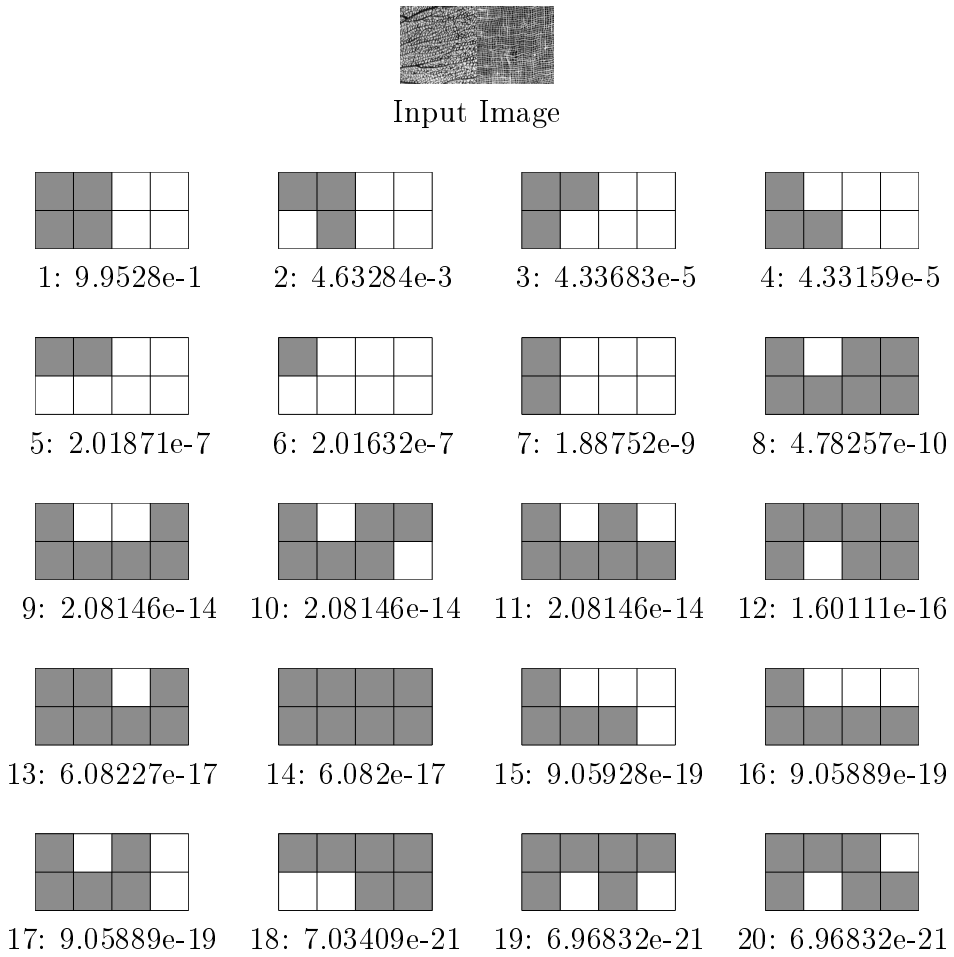
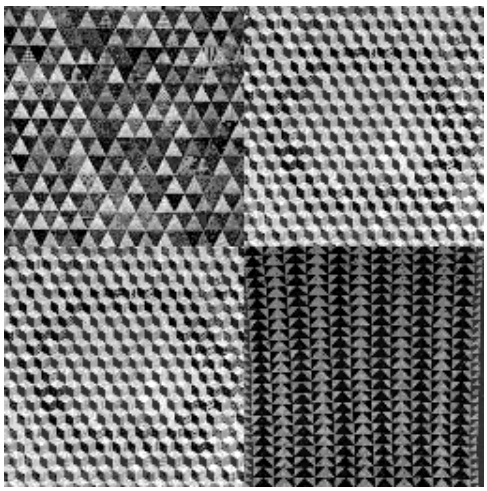


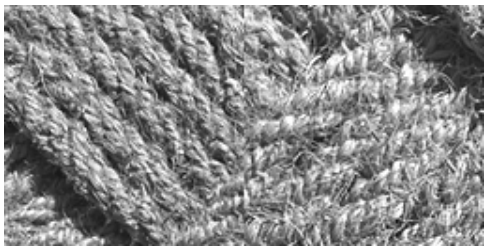
Figure 7.9 The 20 top segments. The region size is 64. There are 23 models. Image filename: brod13-64-23.

7.3.2 Unfamiliar texture mosaics

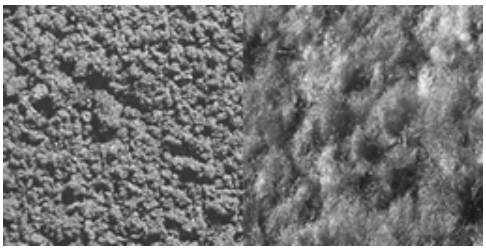
The second category of experiments dealt with texture mosaics that were similar to the images used in Section 7.3.1. The primary difference is that the textures involved in this series of experiments were not in any way used in the generation of class representatives for the algorithm. The hypothesis being examined here is that the textures being used in these experiments are grouped in the space of possible class representatives. That is, even though these particular textures have not been seen before, the textures that have been seen (during the generation of the class representatives) will provide enough discrimination to segment the images. This set of experiments supports this idea by showing that the algorithm can discriminate textures that are unfamiliar to the algorithm. Figure 7.10 shows the images used in these experiments. Figures 7.11, 7.12, and 7.13 show segment distributions produced from these images.



qpatch1



F1



F3

Figure 7.10 Test data for Subsection 7.3.2.

Input Image

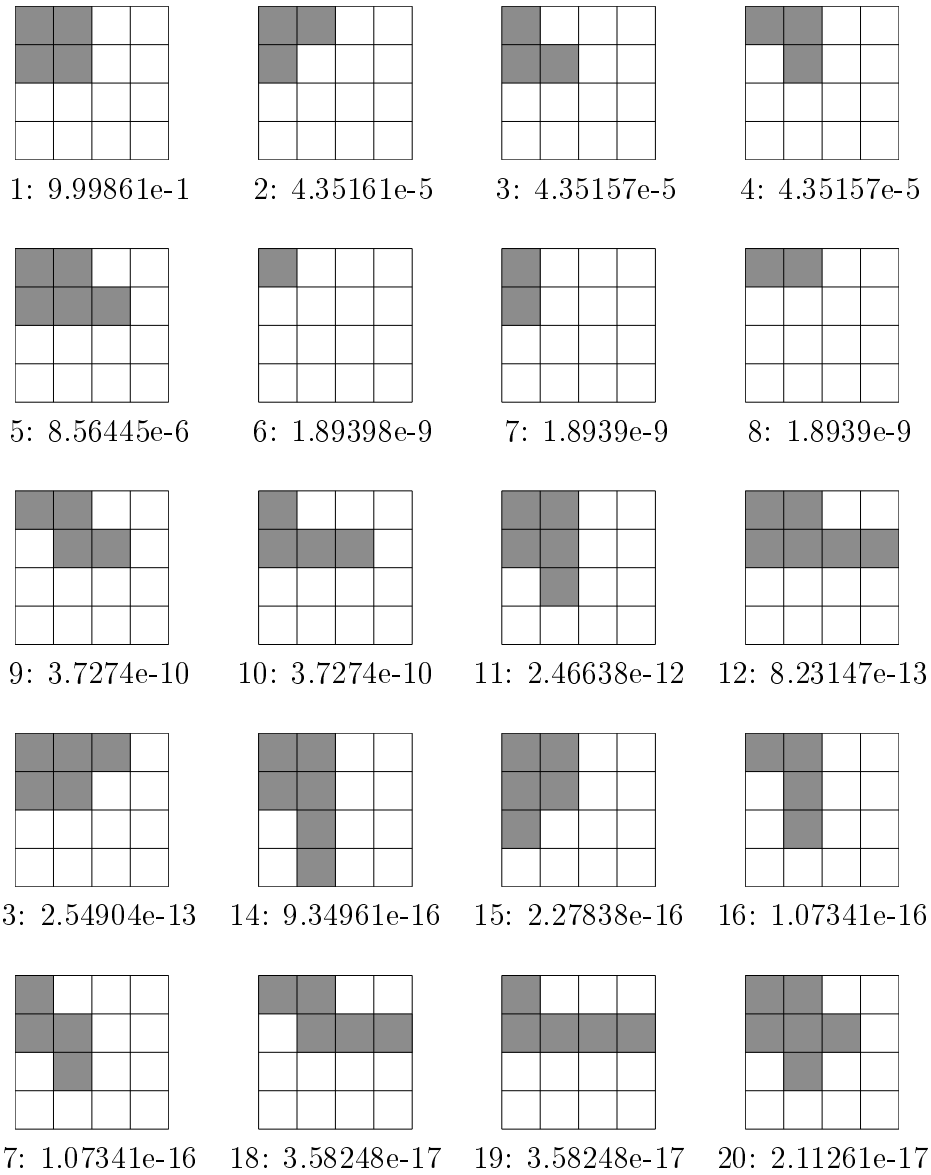


Figure 7.11 The 20 top segments. The region size is 64. There are 23 models. Image filename: qpatch2-64-23.

Input Image

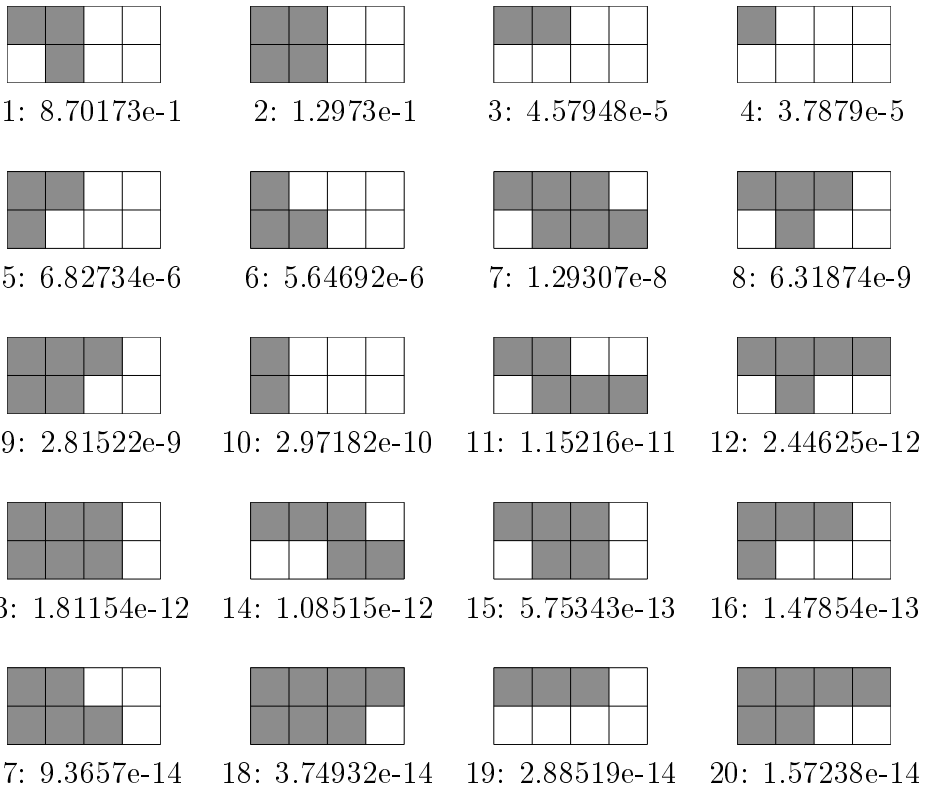
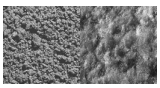


Figure 7.12 The 20 top segments. The region size is 64. There are 23 models. Image filename: F1-64-23.



Input Image

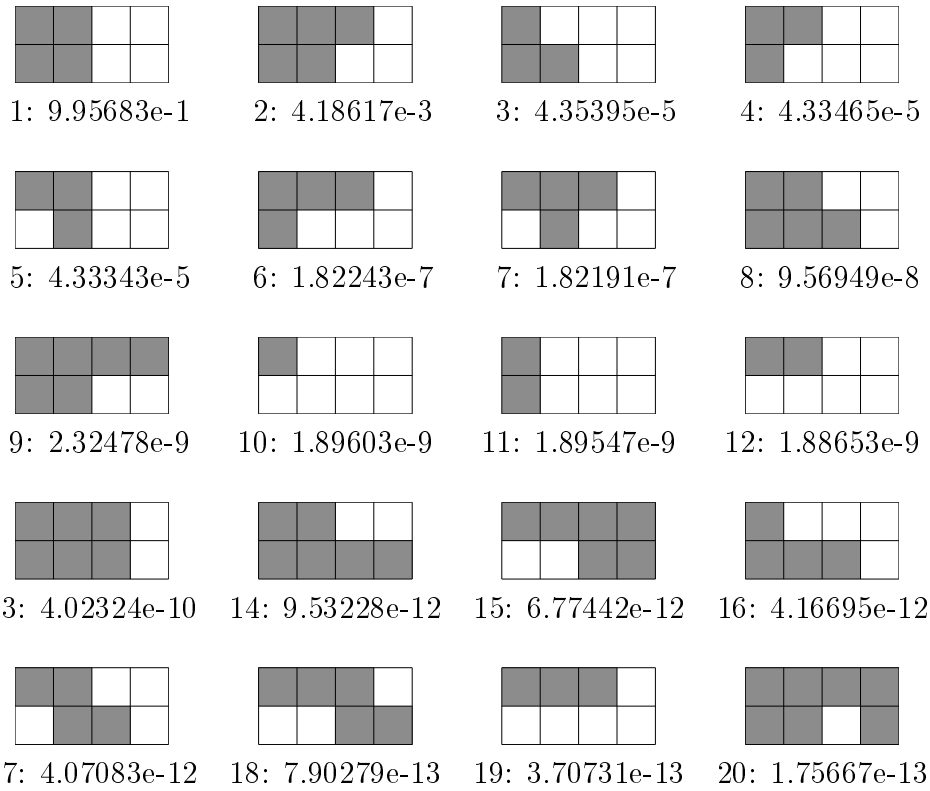


Figure 7.13 The 20 top segments. The region size is 64. There are 23 models. Image filename: F3-64-23.

7.3.3 Natural scenes

The final set of experiments involving distributions of segments examines the algorithm's performance on more natural scenes. Figure 7.14 shows the images used in these experiments. Not surprisingly, the performance on these images, shown in Figures 7.15, 7.16, 7.17, and 7.18, was not as good as seen in the previous two sets of experiments. However, the algorithm does a reasonable job of finding natural textures with enough high-frequency content, such as the grass in the upper portion of Figure 7.18. The performance on portions of images with little high frequency information, such as the sky in Figure 7.15, was predictably poorer.

F5

F7

Figure 7.14 Test data for Subsection 7.3.3.

Input Image

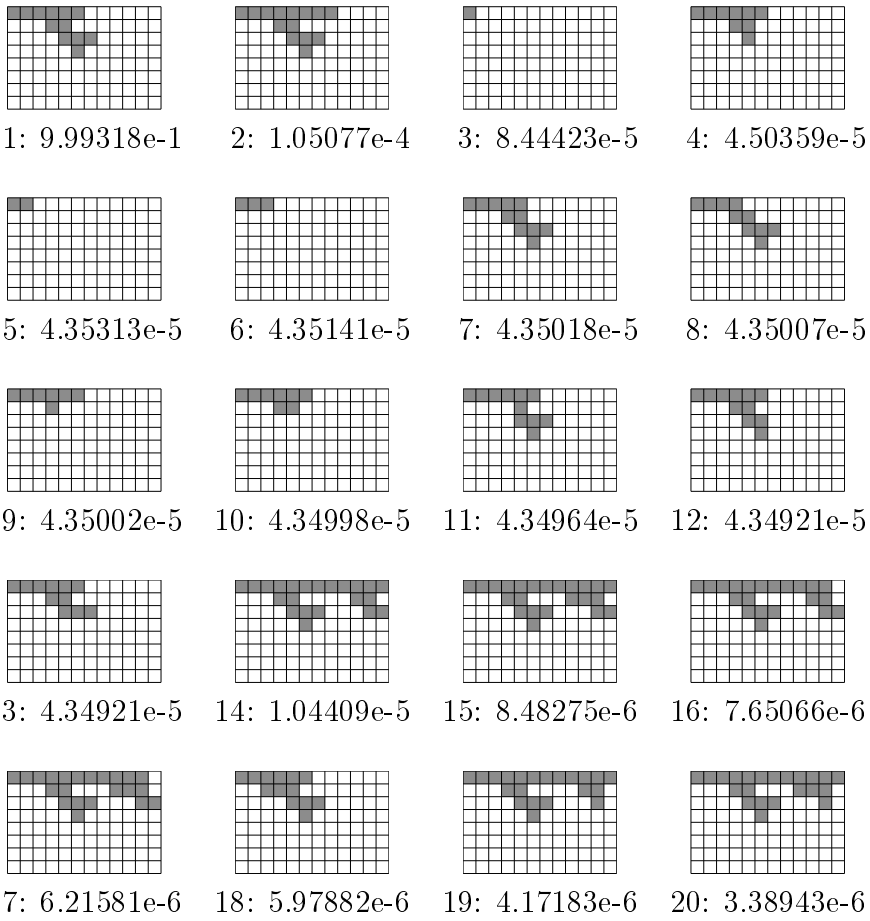


Figure 7.15 The 20 top segments. The region size is 64. There are 23 models. Image filename: F5-64-23. The initial region was on the top row, first column (Region 0).

Input Image

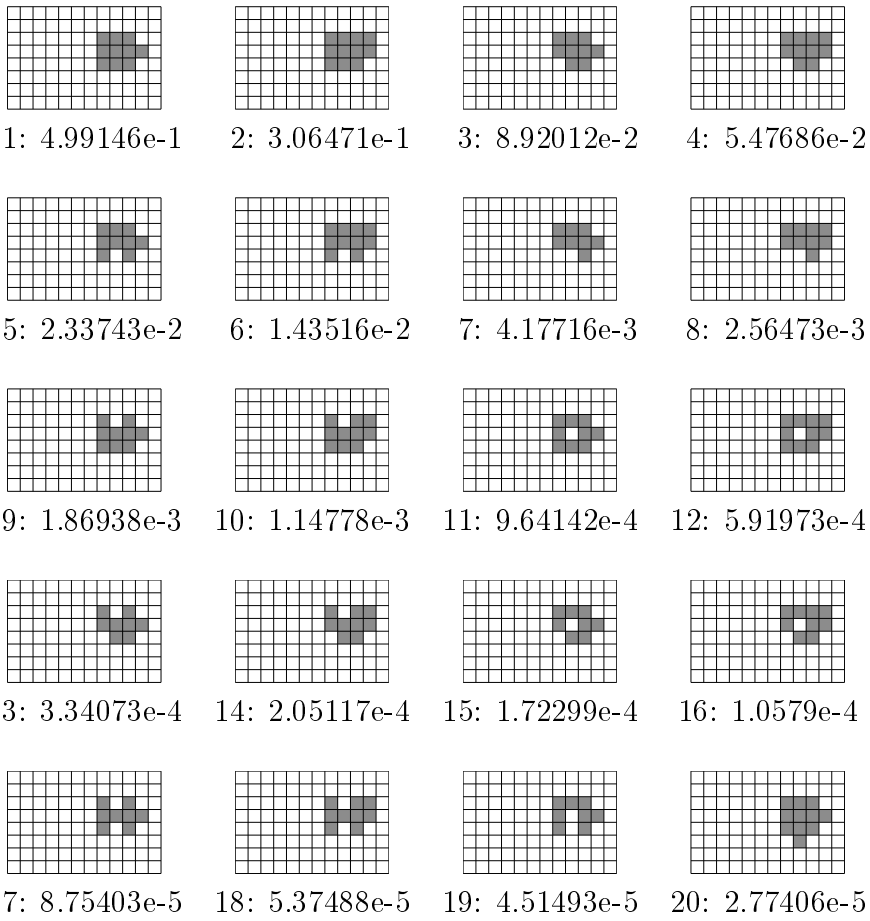


Figure 7.16 The 20 top segments. The region size is 64. There are 23 models. Image filename: F5-64-23. The initial region was on the third row, eighth column (Region 58).

Input Image

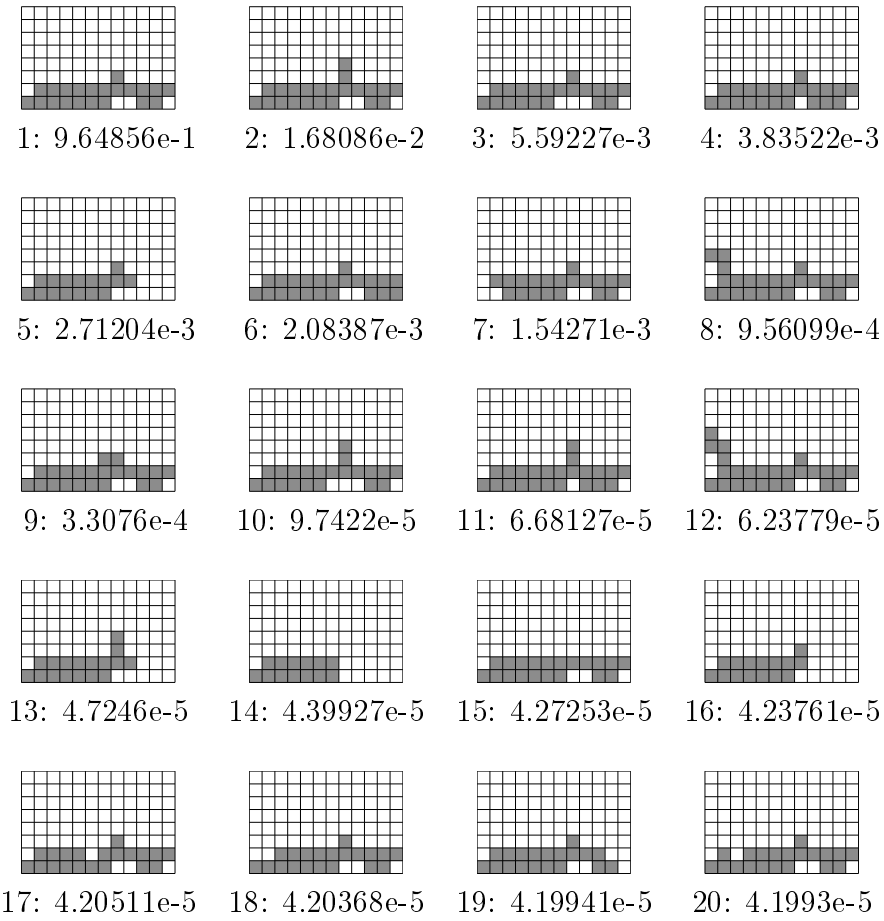
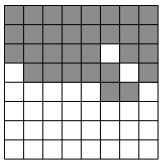
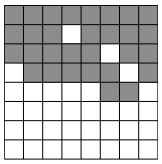


Figure 7.17 The 20 top segments. The region size is 64. There are 23 models. Image filename: F5-64-23. The initial region was on the eighth row, eighth column (Region 63).

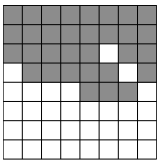
Input Image



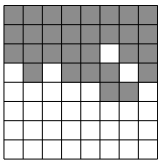
1: 9.98485e-1



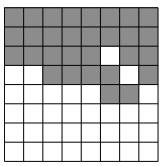
2: 1.38038e-4



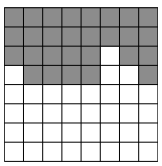
3: 6.36602e-5



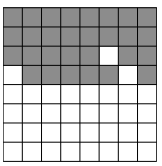
4: 5.95989e-5



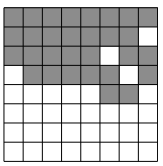
5: 5.69088e-5



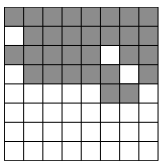
6: 5.4543e-5



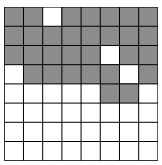
7: 4.91384e-5



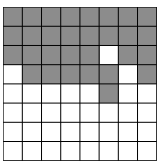
8: 4.65494e-5



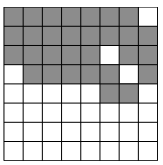
9: 4.52906e-5



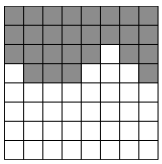
10: 4.40632e-5



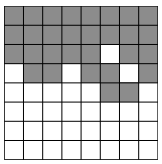
11: 4.35708e-5



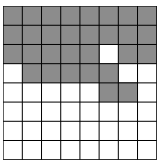
12: 4.35373e-5



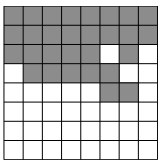
13: 4.34662e-5



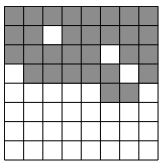
14: 4.34662e-5



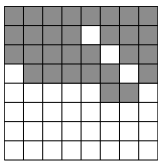
15: 4.34607e-5



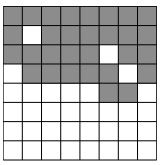
16: 4.34578e-5



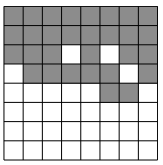
17: 4.34572e-5



18: 4.34566e-5



19: 4.34565e-5



20: 4.34564e-5

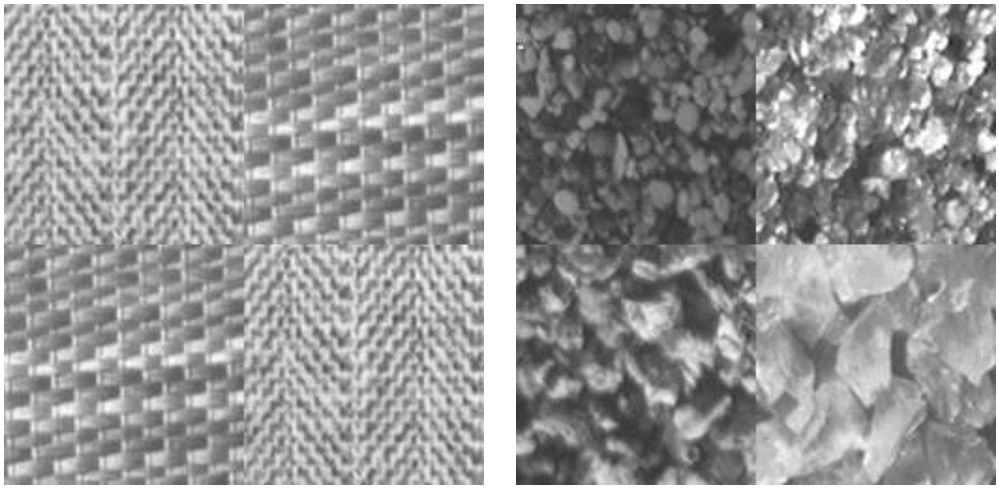
Figure 7.18 The 20 top segments. The region size is 64. There are 23 models. Image filename: F7-64-23.

7.4 Segmentation Distributions

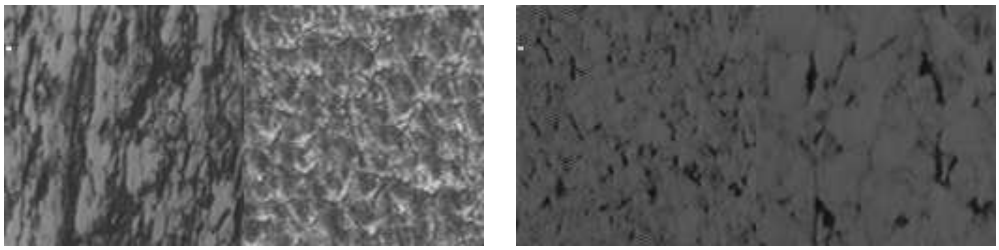
Now we turn to full segmentations and compute distributions of segmentations for the same types of images used above. Recall that the algorithm was tested on Brodatz textures, unfamiliar textures, and natural scenes.

7.4.1 Brodatz textures

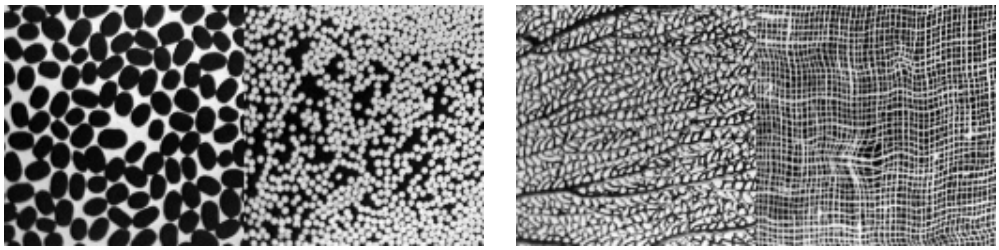
The texture mosaics used in this set of experiments were the same ones used in Section 7.3.1. Again, this set of experiments gives the most favorable results because of two factors: first, the textures are homogeneous and contain mostly high-frequency information, and second, each texture was used in the generation of class representatives for the algorithm. Figure 7.19 shows the images used in these experiments. Figures 7.20, 7.21, 7.22, 7.23, 7.24, and 7.25 show segmentation distributions produced from these images.



brod2 brod4

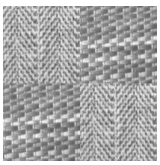


brod7 brod9



brod12 brod13

Figure 7.19 Test data for Subsection 7.4.1.



Input Image

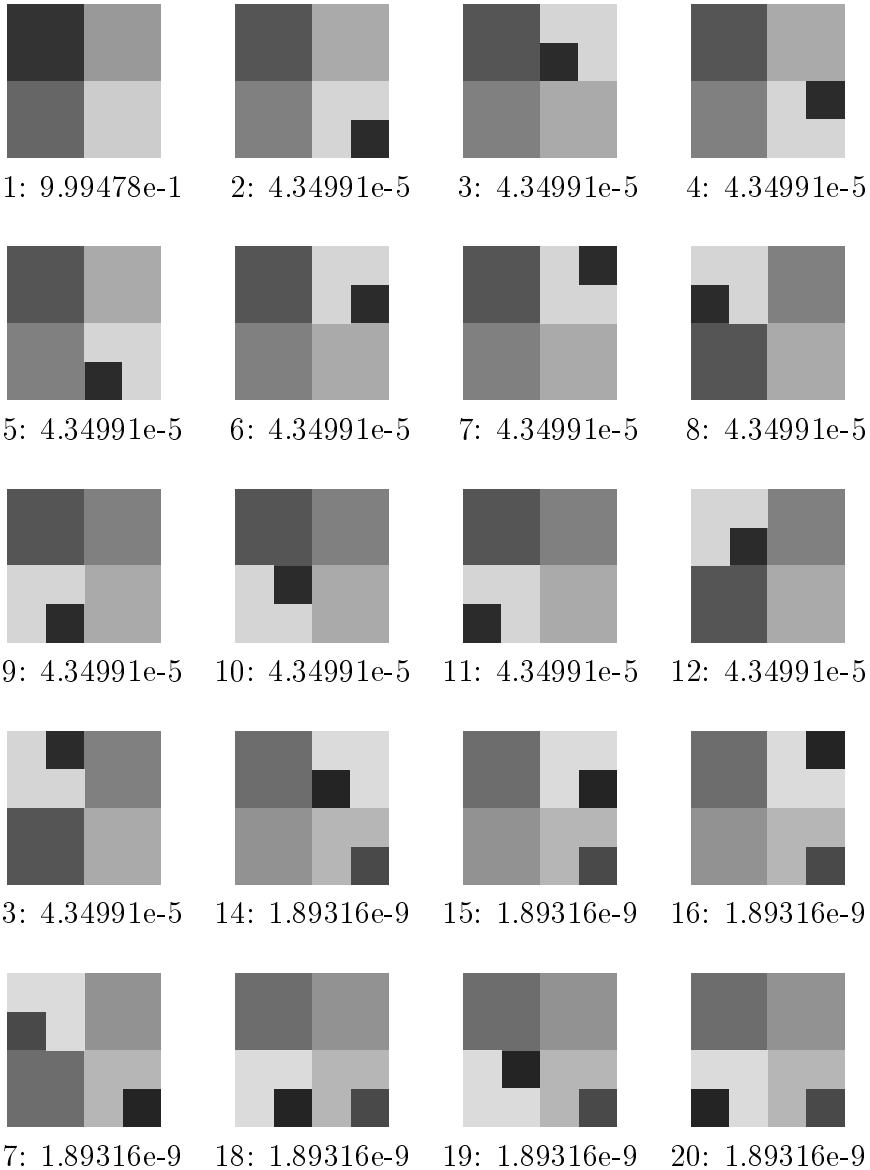


Figure 7.20 The 20 top segmentations. The region size is 64. There are 23 models. Image filename: brod1-64-23.

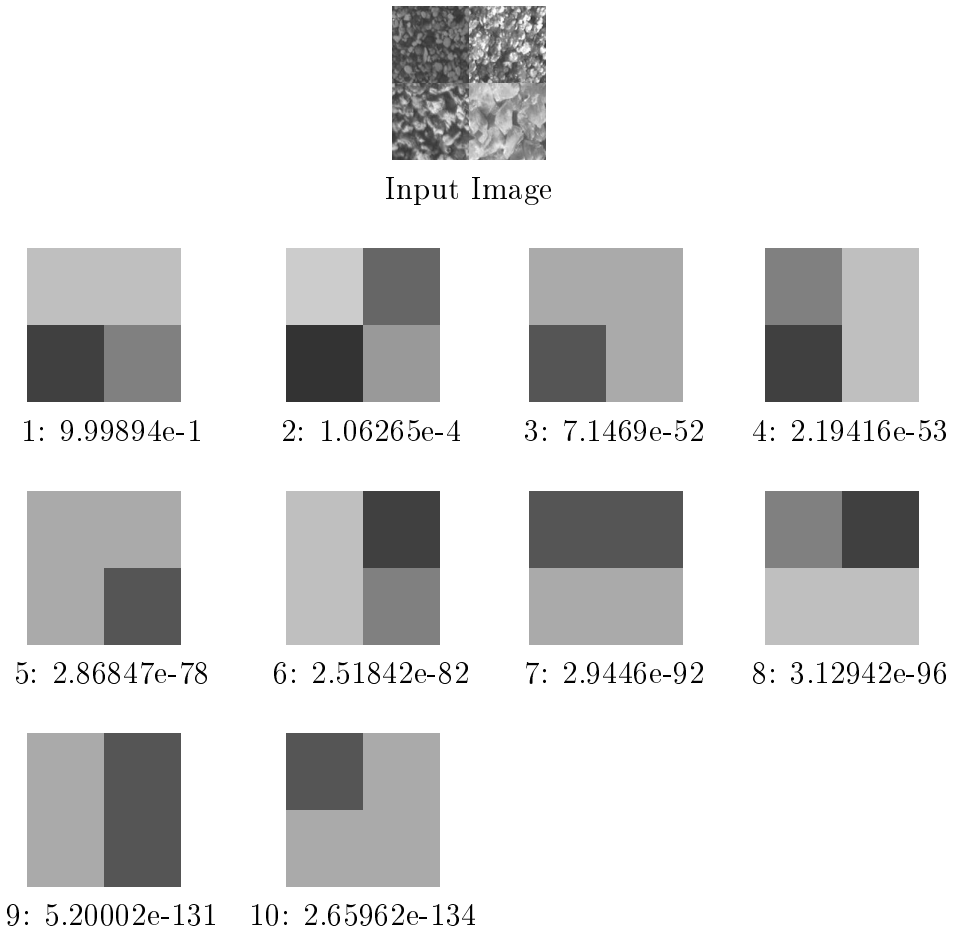


Figure 7.21 The 10 top segmentations. The region size is 128. There are 23 models. Image filename: brod4-128-23.

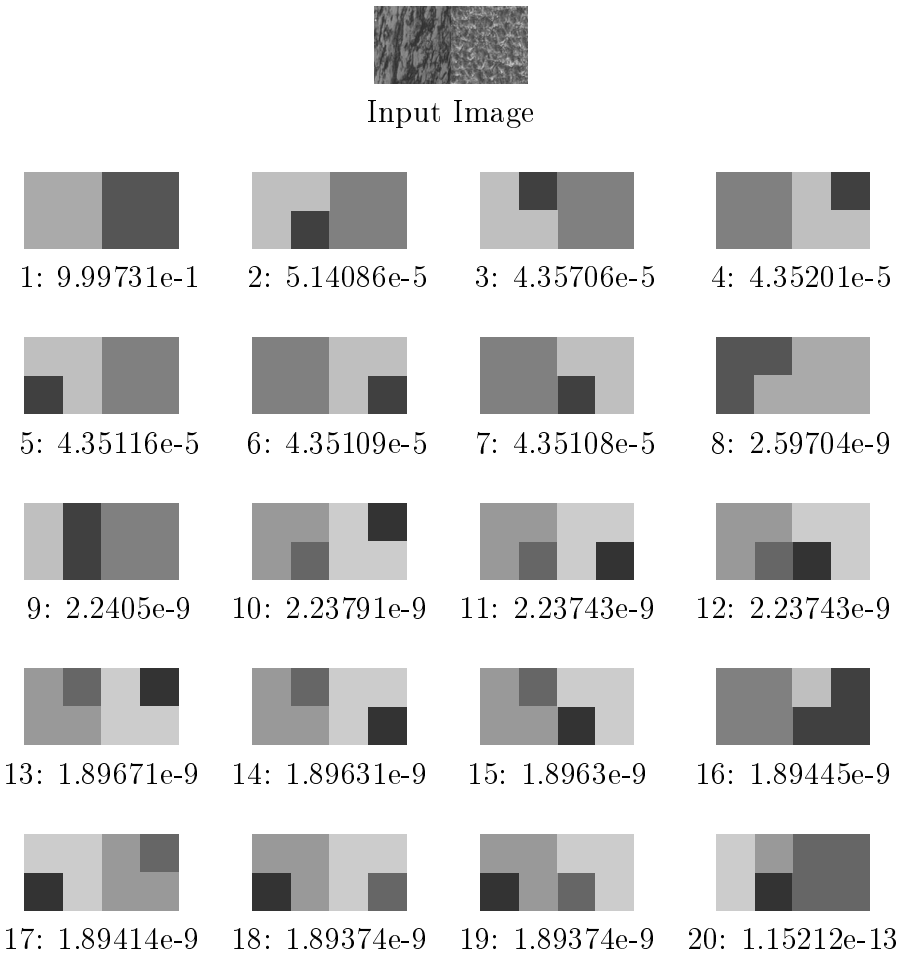


Figure 7.22 The 20 top segmentations. The region size is 64. There are 23 models. Image filename: brod7-64-23.

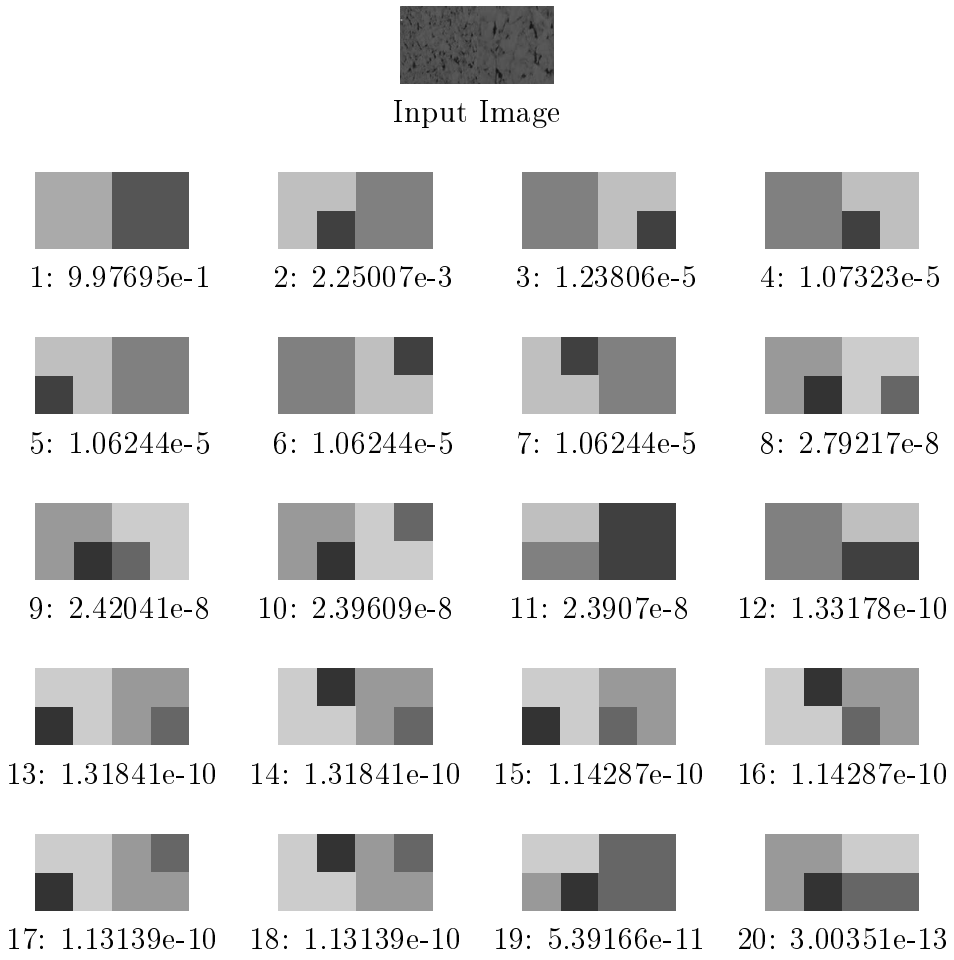


Figure 7.23 The 20 top segmentations. The region size is 64. There are 94 models. Image filename: brod9-64-94.

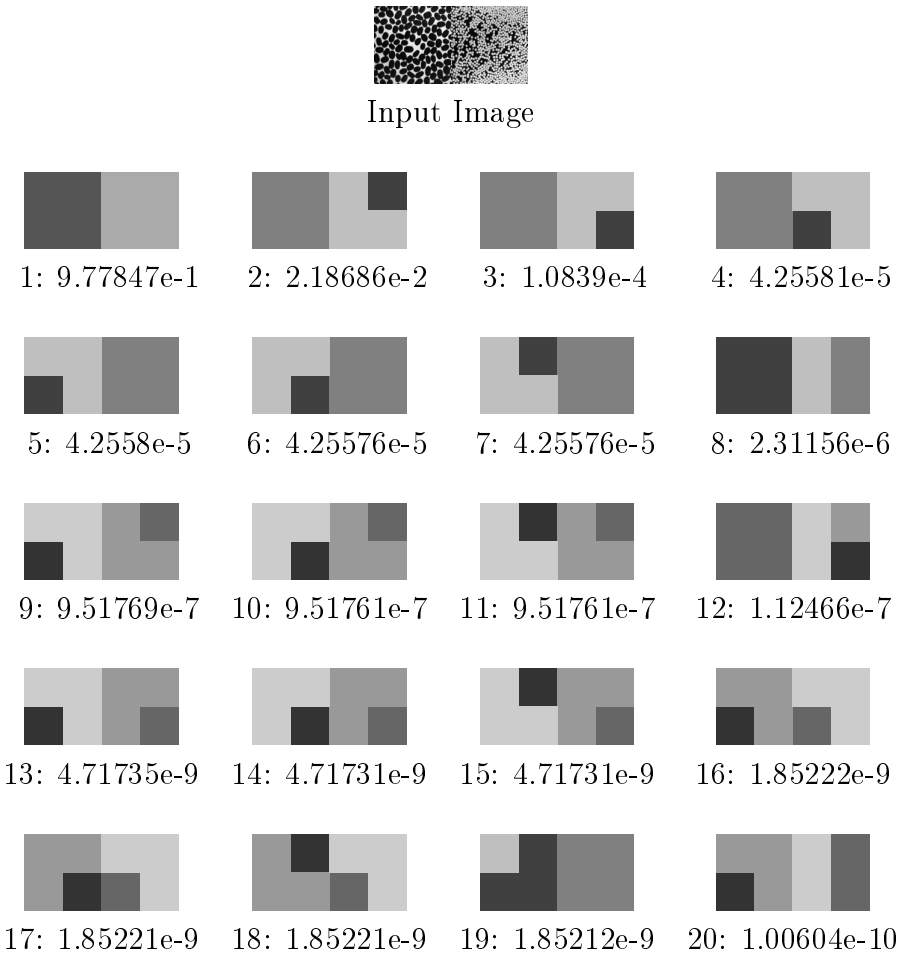


Figure 7.24 The 20 top segmentations. The region size is 64. There are 23 models. Image filename: brod12-64-23.

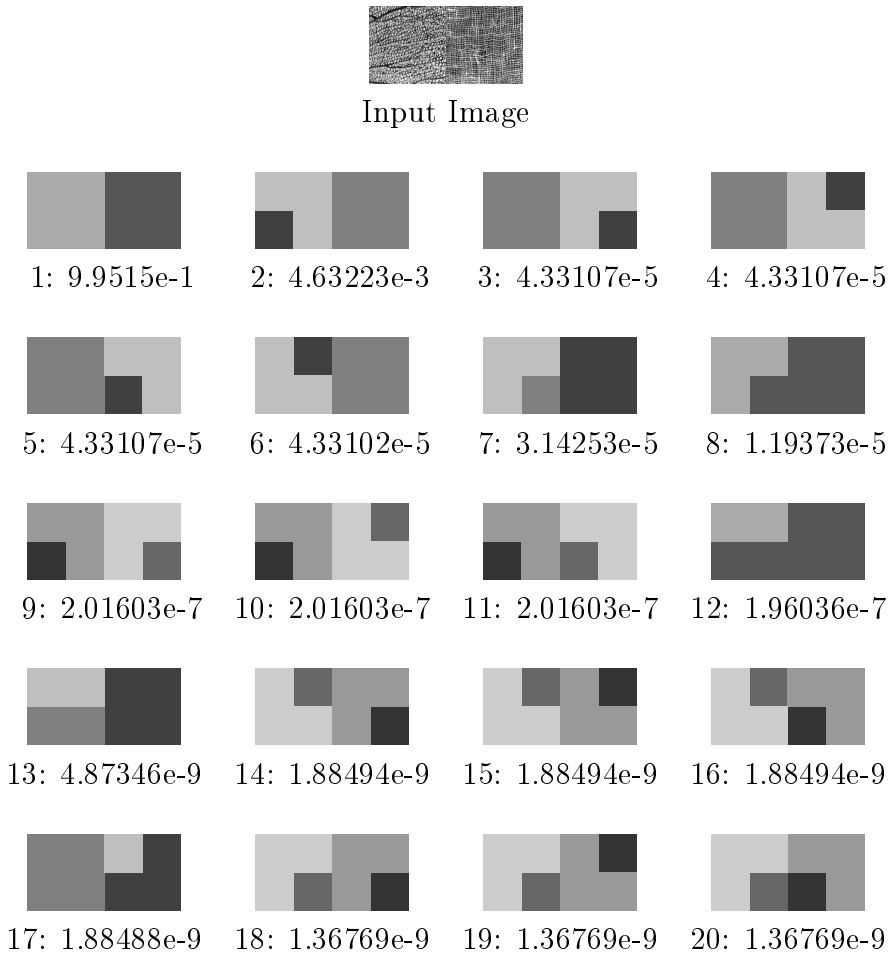
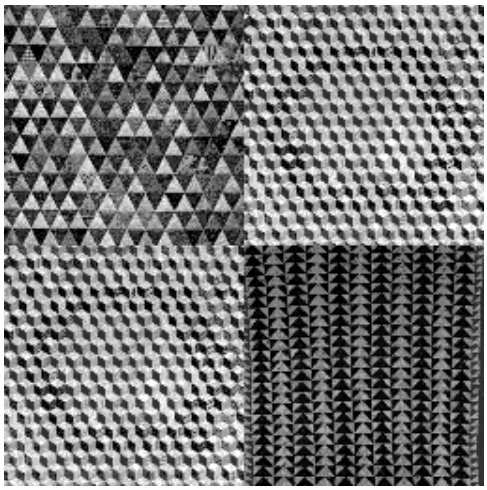
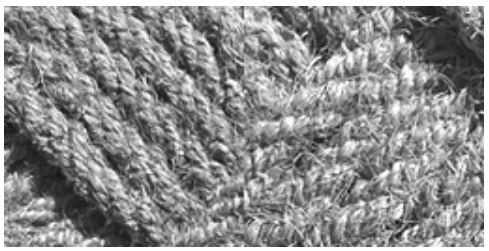


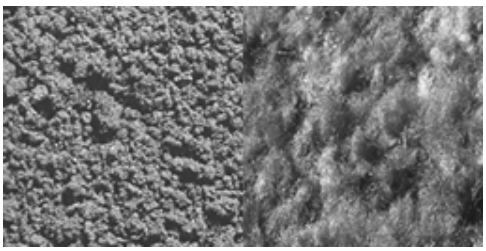
Figure 7.25 The 20 top segmentations. The region size is 64. There are 23 models. Image filename: brod13-64-23.



qpatch2



F1



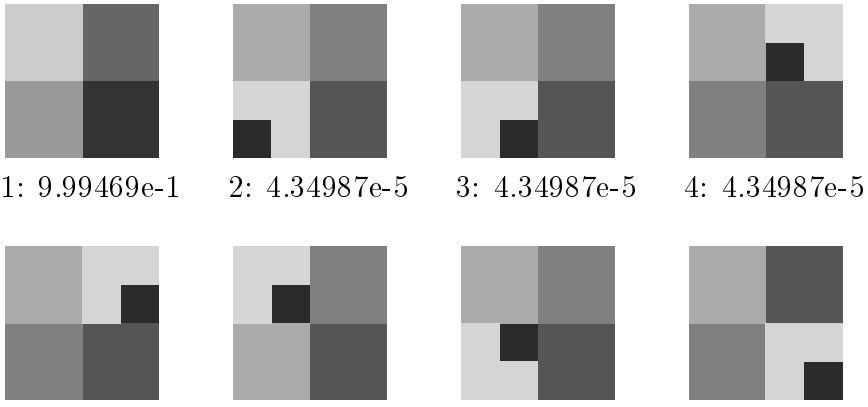
F3

Figure 7.26 Test data for Subsection 7.4.2.

7.4.2 Unfamiliar texture mosaics

This set of experiments concentrated on the same images used in Section 7.3.2. Again, these mostly homogeneous high-frequency textures were not in any way used in the generation of class representatives for the algorithm. Figure 7.26 shows the images used in these experiments. Figures 7.27, 7.28, and 7.29 show segmentation distributions produced from these images.

Input Image



1: 9.99469e-1 2: 4.34987e-5 3: 4.34987e-5 4: 4.34987e-5

5: 4.34987e-5 6: 4.34987e-5 7: 4.34987e-5 8: 4.34987e-5

9: 4.34987e-5 10: 4.34987e-5 11: 4.34987e-5 12: 4.34987e-5

13: 4.34987e-5 14: 8.56147e-6 15: 1.89314e-9 16: 1.89314e-9

17: 1.89314e-9 18: 1.89314e-9 19: 1.89314e-9 20: 1.89314e-9

Figure 7.27 The 20 top segmentations. The region size is 64. There are 23 models. Image filename: qpatch2-64-23.

Input Image

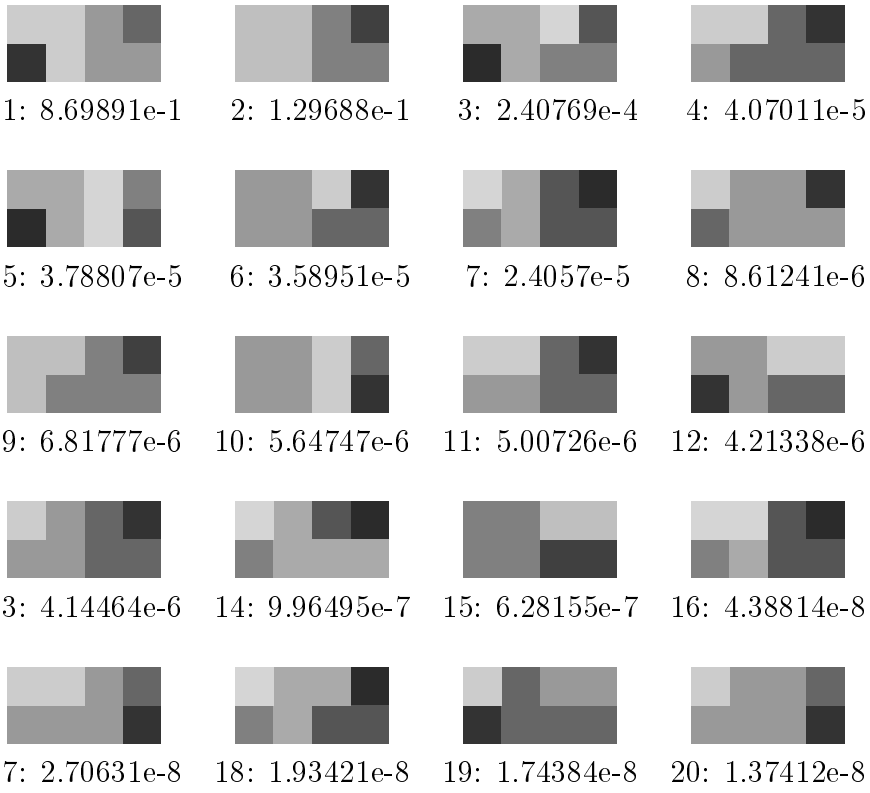


Figure 7.28 The 20 top segmentations. The region size is 64. There are 23 models. Image filename: F1-64-23.

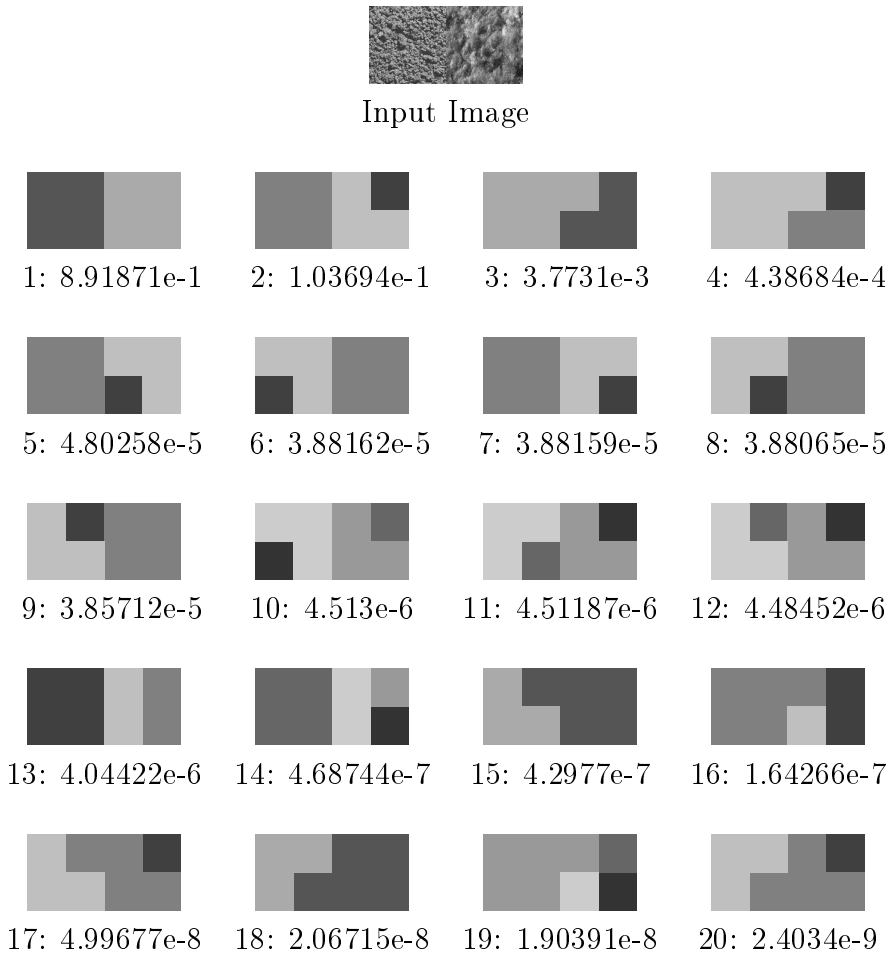
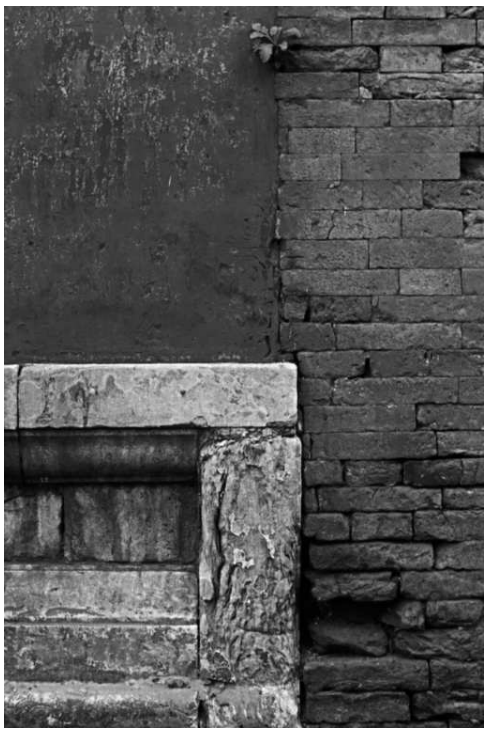


Figure 7.29 The 20 top segmentations. The region size is 64. There are 23 models. Image filename: F3-64-23.

7.4.3 Natural scenes

The final set of experiments involving distributions of segmentations examines the algorithm's performance on more natural scenes, shown in 7.30. Figures 7.31, 7.32, 7.33, 7.34, and 7.35 show that while the performance is not as good as in the previous sections, that the algorithm does tend to group regions containing homogeneous, high-frequency textures. For example, the trees in Figure 7.32 are grouped, since the leaves have what could be considered a homogeneous texture.



F4

F5

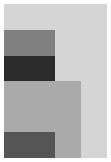
F6

F7

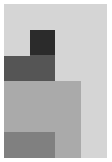
F8

Figure 7.30 Test data for Section 7.4.3.

Input Image



1: 9.99392e-1

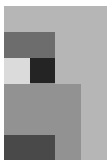


2: 3.97717e-4

3: 1.19599e-5

4: 1.06425e-5

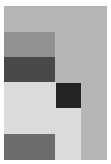
5: 1.06425e-5



6: 1.06425e-5

7: 1.06425e-5

8: 1.06425e-5



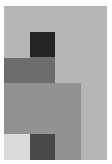
9: 1.06425e-5

10: 1.06425e-5

11: 1.06425e-5

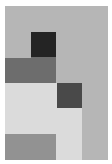
12: 1.06425e-5

13: 4.23527e-9

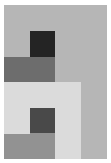


14: 4.23527e-9

15: 4.23527e-9



16: 4.23527e-9



17: 4.23527e-9

18: 4.23527e-9

19: 1.41059e-10

20: 1.2736e-10

Figure 7.31 The 20 top segmentations. The region size is 128. There are 94 models. Image filename: F4-128-94.

Input Image

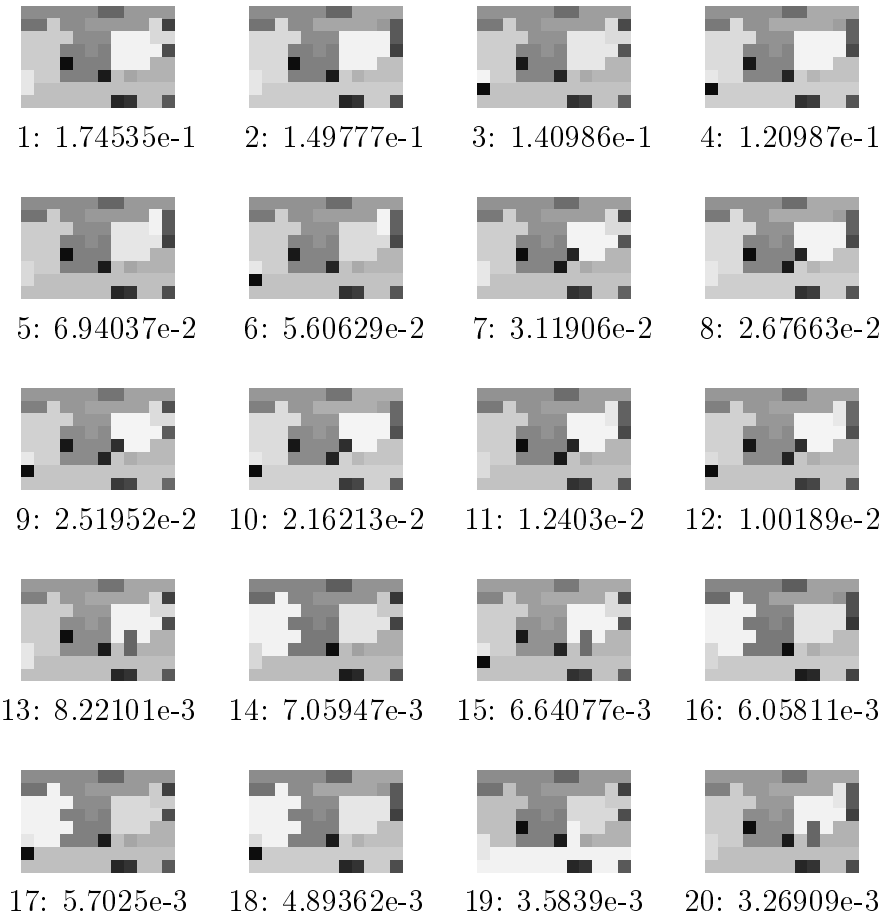


Figure 7.32 The 20 top segmentations. The region size is 64. There are 23 models. Image filename: F5-64-23.

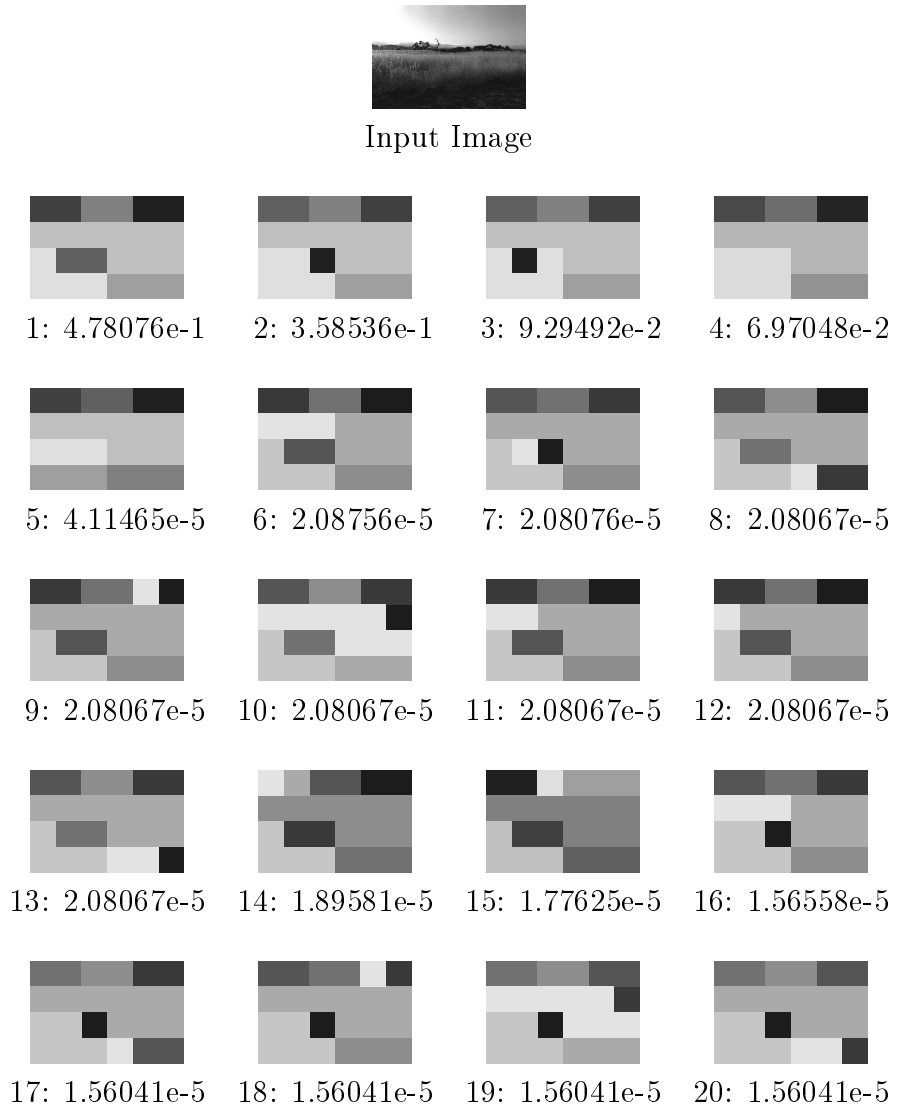


Figure 7.33 The 20 top segmentations. The region size is 128. There are 23 models. Image filename: F6-128-23.

Input Image

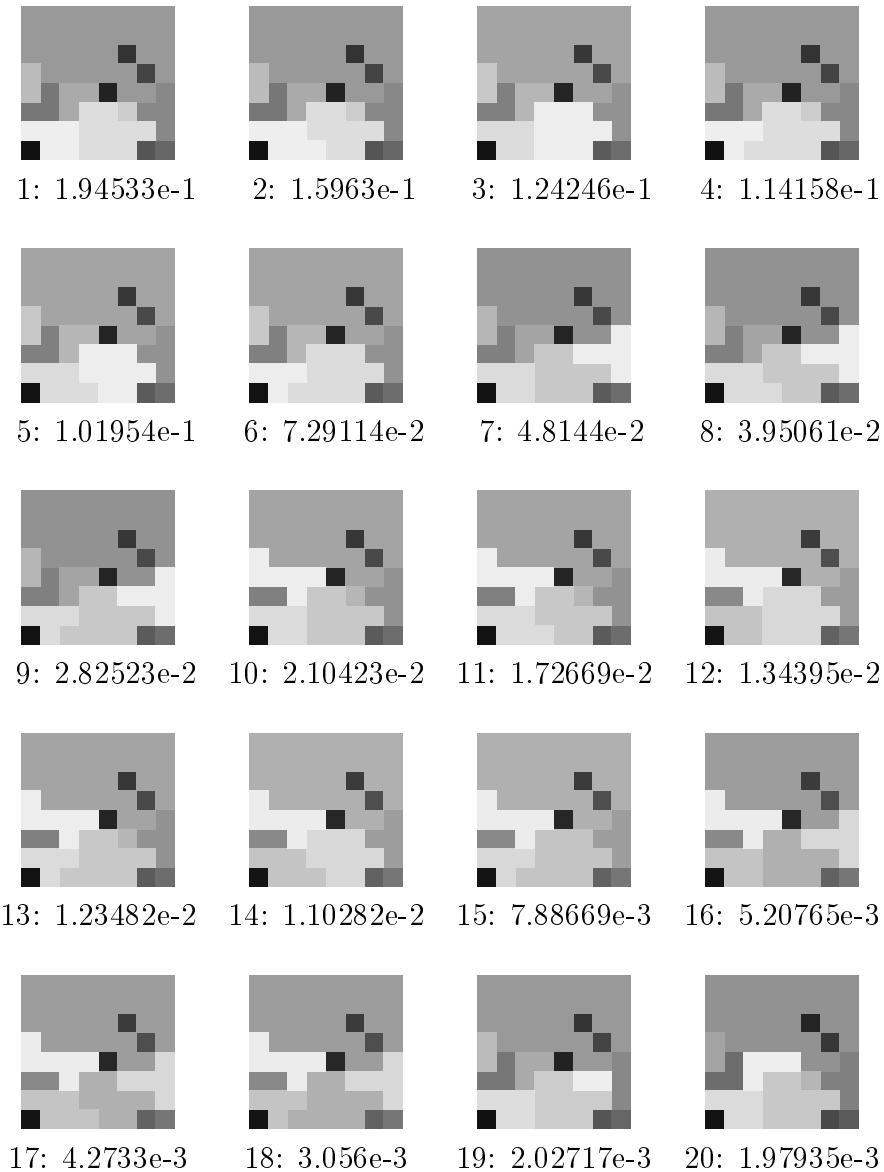


Figure 7.34 The 20 top segmentations. The region size is 64. There are 23 models. Image filename: F7-64-23.

Input Image

1: 5.46655e-1

2: 3.57446e-1

3: 3.16368e-2

4: 2.06867e-2

5: 1.25224e-2

6: 8.18812e-3

7: 6.97229e-3

8: 4.55903e-3

9: 2.59355e-3

10: 1.69587e-3

11: 1.25207e-3

12: 4.20748e-4

13: 4.0351e-4

14: 3.92679e-4

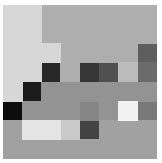
15: 2.75118e-4

16: 2.63847e-4

17: 2.56764e-4

18: 1.59716e-4

19: 1.50098e-4



20: 1.04435e-4

Figure 7.35 The 20 top segmentations. The region size is 64. There are 23 models. Image filename: F8-64-23.

CHAPTER 8

EXTENSIONS AND CONCLUSIONS

8.1 Extensions

Much more work remains to be done in the area of image segmentation. The field of modeling texture with mathematical models, in particular, is rife with opportunity. Much work has been done in this field (see Chapter 2 for an overview), but the results of these models on natural textures show that there is room for improvement [14], [21], [24].

Since we have seen that the third-order MRF model doesn't model natural texture well, a possible direction would be to investigate more complex models for texture. For example, higher-order MRF models could be used.

A more promising area for further research is consideration of multiple models. Chapter 7 showed some instances where using a third-order MRF does not model the texture in the image well. Incorporating information from texture models that do well where a third-order MRF model fails could improve the reliability of this algorithm.

Also of interest is how domain knowledge could influence the prior distribution in the Bayesian analysis. To go back to the example of Chapter 3, if a facet model is in use and it is known that all planes meet at right angles, (3.6) could be modified to take

this into account. Similarly, if images of newspapers are under consideration, certain assumptions about newspapers, such as the texture of news print or the fact that text usually is parallel to the edge of the page, can be incorporated into this analysis.

8.2 Conclusion

We have extended the probabilistic framework of LaValle [46] for considering distributions of segmentations of textured images. A simple probabilistic model for modeling error and noise has been presented.

Although a third-order Markov random field is not a good model for naturally occurring textures, this framework gives us the ability to return a set of possible solutions to the segmentation problem, along with a confidence measure for possible use by a higher-level process. Both the individual values and information about the distribution of those values are important. For example, if the model used to evaluate the data gives little information, the shape of the confidence measures tends to be flat (all solutions returned have approximately equal probabilities). Modeling uncertainty in low-level computer vision and communicating this information along with a solution are ideas which have begun to be stressed as important [49], [69], and represent promising directions for the integration of low-level and mid-level computer vision algorithms.

REFERENCES

[1] J. Beaulieu and M. Goldberg. Hierarchy in picture segmentation: a stepwise optimization approach. *IEEE Trans. Pattern Anal. Mach. Intell.*, 11(2):150–163, February 1989.

[2] T. Bell. Remote sensing. *IEEE Spect.*, 32(3):24–31, March 1995.

[3] D. A. Berry, I. W. Evitt, and R. Pinchin. Statistical inference in crime investigations using deoxyribonucleic acid profiling. *J. R. Stat. Soc.*, C41:499–531, 1992.

[4] P. J. Besl and R. C. Jain. Segmentation through variable-order surface fitting. *IEEE Trans. Pattern Anal. Mach. Intell.*, 10(2):167–191, March 1988.

[5] J. R. Beveridge, J. Griffith, R. R. Kohler, A. R. Hanson, and E. M. Riseman. Segmenting images using localized histograms and region merging. *Int. J. Comput. Vis.*, 2(3):311–347, January 1989.

[6] J. Bigün and J. M. H. du Buf. N-folded symmetries by complex moments in Gabor space and their application to unsupervised texture segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 16(1):80–87, January 1994.

[7] C. Bouman and B. Liu. Multiple resolution segmentation of textured images. *IEEE Trans. Pattern Anal. Mach. Intell.*, 13(2):99–113, February 1991.

[8] P. Brodatz. *Textures; a photographic album for artists and designers.* Dover Publications, New York, NY, 1966.

[9] R. Brualdi. *Introductory Combinatorics.* Elsevier Science Publishing Co., Inc., New York, NY, 1977.

[10] R. L. Castaño. A probabilistic framework for grouping image features. M. S. Thesis, University of Illinois, Urbana, December 1994.

[11] K. P. S. Chan and C. G. G. Aitkin. Estimation of the Bayes Factor in a forensic science problem. *J. Stat. Comput. Simul.*, 33:249–264, 1991.

[12] R. Chellappa and S. Chatterjee. Classification of textures using Gaussian Markov Random Fields. *IEEE Trans. Acoust. Speech Signal Process.*, 33(4):959–963, August 1985.

[13] C. Chen and R. Dubes. Discrete MRF model parameters as features for texture classification. *Proc. IEEE Int. Conf. Syst. Man Cybern.*, 1990.

[14] C.C. Chen and R.C. Dubes. Experiments in fitting discrete Markov random fields to textures. *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 89:298–303, 1989.

[15] C. Chu and J. K. Aggarwal. The integration of region and edge-based segmentation. In *Proc. Int. Conf. Comput. Vis.*, pages 117–120, 1990.

[16] J. Coggins and A. Jain. A spatial filtering approach to texture analysis. *Pattern Recognit. Lett.*, 3(3):195–203, May 1985.

[17] F. Cohen, Z. Fan, and S. Attali. Automated inspection of textile fabrics using textural models. *IEEE Trans. Pattern Anal. Mach. Intell.*, 13(8):803–808, August 1991.

[18] F. Cohen, Z. Fan, and M. Patel. Classification of rotated and scaled textured images using Gaussian Markov random field models. *IEEE Trans. Pattern Anal. Mach. Intell.*, 13(2):192–202, February 1991.

[19] F. S. Cohen and D. B. Cooper. Simple parallel hierarchical and relaxation algorithms for segmenting noncausal Markovian random fields. *IEEE Trans. Pattern Anal. Mach. Intell.*, 9(2):195–219, March 1987.

[20] F. S. Cohen and Z. Fan. Maximum likelihood unsupervised textured image segmentation. *Comput. Vis. Graph. Image Process.*, 54(3):239–251, May 1992.

[21] R. Conners and C. Harlow. A theoretical comparison of texture algorithms. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2(3):204–222, May 1980.

[22] M. Daily. Color image segmentation using Markov random fields. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, pages 304–312, June 1989.

[23] H. Derin and H. Elliott. Modeling and segmentation of noisy and textured images using Gibbs random fields. *IEEE Trans. Pattern Anal. Mach. Intell.*, 9(1):39–55, January 1987.

[24] J. M. H. du Buf, M. Kardan, and M. Spann. Texture feature performance for image segmentation. *Pattern Recognit.*, 23:291–309, 1990.

[25] R. C. Dubes, A. K. Jain, S. G. Nadabar, and C. C. Chen. MRF model-based algorithms for image segmentation. In *Proc. IEEE 10th Int. Conf. Pattern Recognit.*, pages 808–814, Atlantic City, NJ, June 1990.

[26] R. Duda and P. Hart. *Pattern Classification and Scene Analysis.* John Wiley & Sons, Inc., New York, NY, 1972.

[27] D. Dunn, W. Higgins, and J. Wakeley. Texture segmentation using 2-d Gabor elementary functions. *IEEE Trans. Pattern Anal. Mach. Intell.*, 16(2):130–149, February 1994.

[28] I. Elfadel and R. Picard. Gibbs random fields, co-occurrences, and texture modeling. *IEEE Trans. Pattern Anal. Mach. Intell.*, 16(1):24–37, January 1994.

[29] Z. Fan and F. Cohen. Textured image segmentation as a multiple hypothesis test. *IEEE Trans. Circuits Syst.*, 35(6):691–702, June 1988.

[30] K. S. Fu, R. C. Gonzalez, and C. S. G. Lee. *Robotics: Control, Sensing, Vision, and Intelligence*. McGraw-Hill, New York, NY, 1987.

[31] D. Gabor. Theory of communication. *J. Inst. Elect. Engr.*, 93:429–457, 1946.

[32] D. Geman and S. Geman. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Trans. Pattern Anal. Mach. Intell.*, 6(6):721–741, November 1984.

[33] A. Gray, J. Kay, and D. Titterington. An empirical study of the simulation of various models used for images. *IEEE Trans. Pattern Anal. Mach. Intell.*, 16(5):507–513, May 1994.

[34] J. F. Haddon and J. F. Boyce. Image segmentation by unifying region and boundary information. *IEEE Trans. Pattern Anal. Mach. Intell.*, 12(10):929–948, October 1990.

[35] R. M. Haralick and L. Watson. A facet model for image data. *Comput. Vis. Graph. Image Process.*, 15:113–129, February 1981.

[36] R.M. Haralick, K. Shanmugam, and I. Dinstein. Textural features for image classification. *IEEE Trans. Syst. Man Cybern.*, 3:660–671, November 1971.

[37] B. K. P. Horn. *Robot Vision*. MIT Press, Cambridge, MA, 1986.

[38] E. Ising. Beitrag zur theorie des ferromagnetismus. *Zeitschrift fur Physik*, 31:253–258, Apr 1925.

[39] A. Jain and S. Bhattacharjee. Text segmentation using Gabor filters for automatic document processing. *Mach. Vis. Appl.*, 5:169–184, 1992.

[40] A. Jain and S. Nadabar. MRF model-based segmentation of range images. In *Proc. Int. Conf. Comput. Vis.*, 1990.

[41] A. Jain and S. Nadabar. MRF applications in image analysis. *Data Analy. Astronomy IV*, 1992.

[42] R. L. Kashyap and R. Chellappa. Estimation and choice of neighbors in spatial-interaction models of images. *IEEE Trans. Inf. Theory*, 29(1):60–72, January 1983.

[43] A. Khotanzad and J. Chen. Unsupervised segmentation of textured images by edge detection in multidimensional features. *IEEE Trans. Pattern Anal. Mach. Intell.*, 11(4):414–421, April 1989.

[44] S. Kollias and L. Sukissian. Adaptive segmentation of textured images using linear prediction and neural networks. *Neural Networks Signal Process.: Proc. IEEE Signal Process. Workshop*, 1992.

[45] S. Lakshmanan and H. Derin. Simultaneous parameter estimation and segmentation of Gibbs random fields using simulated annealing. *IEEE Trans. Pattern Anal. Mach. Intell.*, 11(48):799–813, August 1989.

[46] S. M. LaValle. A Bayesian framework for considering probability distributions of image segments and segmentations. M. S. Thesis, University of Illinois, Urbana, December 1992.

[47] S. M. Lavalle and S. A. Hutchinson. A Bayesian segmentation methodology for parametric image models. Technical Report UIUC-BI-AI-RCV-93-06, Beckman Institute, University of Illinois, August 1993.

[48] S. M. Lavalle and S. A. Hutchinson. A Bayesian segmentation methodology for parametric image models. *IEEE Trans. Pattern Anal. Mach. Intell.*, 17(2):211–217, February 1995.

[49] S. M. Lavalle and S. A. Hutchinson. A framework for constructing probability distributions on the space of image segmentations. *Comput. Vis. Image Understanding*, 61(2):203–230, March 1995.

[50] K. Laws. Textured image segmentation. Technical Report USCIPI Report 940, Dept. of Elec. Eng., Image Processing Institute, University of Southern California, Los Angeles, January 1980.

[51] A. Leonardis, A. Gupta, and R. Bajcsy. Segmentation as the search for the best description of the image in terms of primitives. In *Proc. Int. Conf. Comput. Vis.*, pages 121–125, 1990.

[52] L. M. Lifshitz and S. M. Pizer. A multiresolution hierarchical approach to image segmentation based on intensity extrema. *IEEE Trans. Pattern Anal. Mach. Intell.*, 12(6):529–540, June 1990.

[53] F. Liu and R. Picard. Periodicity, directionality, and randomness: Wold features for perceptual pattern recognition. *Proc. Int. Conf. Pattern Recognit.*, 94:184–189, 1994.

[54] B. S. Manjunath and R. Chellappa. Unsupervised texture segmentation using Markov random field models. *IEEE Trans. Pattern Anal. Mach. Intell.*, 13(5):478–482, May 1991.

[55] Fl. Marques, J. Cunillera, and A. Gasull. Hierarchical segmentation using compound Gauss-Markov random fields. *Proc. IEEE Conf. Acoust. Speech Signal Process.*, 1992.

[56] D. K. Panjwani and G. Healey. Unsupervised segmentation of textured color images using markov random fields. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, pages 776–777, New York, June 1993.

[57] T. Pavlidis. *Structural Pattern Recognition.* Springer-Verlag, New York, NY, 1977.

[58] A. Pérez and R. C. Gonzalez. An iterative thesholding algorithm for image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 9(6):742–751, November 1987.

[59] A. Perry and D. G. Lowe. Segmentation of textured images. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, pages 319–324, June 1989.

[60] R. Picard, T. Kabir, and F. Liu. Real-time recognition with the entire Brodatz texture database. *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 93:638–639, 1993.

[61] W. Press, B.Flannery, S. Teukolsky, and W. Vetterling. *Numerical Recipes in C.* Cambridge University Press, Cambridge, MA, 2 edition, 1988.

[62] S. J. Raudys and A. K. Jain. Small sample size effects in statistical pattern recognition: recommendations for practitioners. *IEEE Trans. Pattern Anal. Mach. Intell.*, 13(3):252–264, March 1991.

[63] T. R. Reed and J. M. Hans du Buf. A review of recent texture segmentation and feature extraction techniques. *Comput. Vis. Graph. Image Process.*, 57:359–372, May 1993.

[64] T. R. Reed and H. Wechsler. Segmentation of textured images and Gestalt organization using spatial/spatial frequency representations. *IEEE Trans. Pattern Anal. Mach. Intell.*, 12(1):1–12, January 1990.

[65] A. Rosenfeld and A. Kak. *Digital Picture Processing.* Academic Press, New York, NY, 2 edition, 1982.

[66] J. F. Silverman and D. B. Cooper. Bayesian clustering for unsupervised estimation of surface and texture models. *IEEE Trans. Pattern Anal. Mach. Intell.*, 10(4):482–496, July 1988.

[67] A. F. M. Smith and D. J. Speigelhalter. Bayes' factors and choice criteria for linear models. *J. R. Stat. Soc.*, B42:213–220, 1980.

[68] H. Stark and J. Woods. *Probability, Random Processes, and Estimation Theory for Engineers.* Prentice-Hall, Englewood Cliffs, NJ, 1986.

[69] R. Szeliski. Bayesian modeling of uncertainty in low-level vision. *Int. J. Comput. Vis.*, 5(3):271–301, December 1990.

[70] Vision and MIT Media Lab Modeling Group. Vision texture database. http://www-white.media.mit.edu/vismod/imagery/VisionTexture/vistex.html.