Image and Vision Computing, Vol. 15, 1997, pp. 781-795

Textured Image Segmentation: Returning Multiple Solutions

Kevin M. Nickels
knickels@uiuc.edu
Office: (217) 355-1372
Fax: (217) 244-8371

Seth Hutchinson
seth@uiuc.edu
Office: (217) 244-5570
Fax: (217) 244-8371

Dept. of Electrical and Computer Engineering and The Beckman Institute
University of Illinois at Urbana-Champaign
Urbana, IL 61801

Abstract

Traditionally, the goal of image segmentation has been to produce a single partition of an image. This partition is compared to some "ground truth," or human approved partition, to evaluate the performance of the algorithm. This paper utilizes a framework for considering a range of possible partitions of the image to compute a probability distribution on the space 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. The distributions thus returned may be passed to higher-level algorithms to better enable them to interpret the segmentation results. 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.

Keywords: Segmentation, Grouping, Bayesian Methods

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 [2] [11] [28]. Segmentation of an image is a complex but extremely important task in computer vision, usually conceded to be prerequisite to high-level vision algorithms such as object description, recognition, or scene analysis.

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 in an image, 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.

Texture is a popular feature to use when segmenting images. Texture in this context means a pattern of grey scale variation in an intensity image. 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, for example, allow the computer to read the text in a newspaper and to ignore the images when doing optical character recognition [15]. Textiles can be inspected for defects by searching for the inconsistencies in texture caused by a manufacturing problem [7]. Mathematical models for naturally occurring texture are not yet fully developed. For example, many texture models in use today cannot discriminate between some synthetic textures that are visually distinct. Many texture models have difficulty distinguishing between naturally occurring textures as well.

Most classical segmentation algorithms restrict the set of possible solutions by using various constraints to obtain a unique stable solution to the problem [2] [8] [16] [18] [23]. In contrast to this, LaValle and Hutchinson [22] introduce a region-based method that modifies the split-and-merge paradigm to create a set of possible solutions, and returns this set. The method was subsequently extended by Castaño and Hutchinson [4] to the problem of identifying symmetries in intensity images. This new approach allows the representation of every possible image segmentation, either explicitly or implicitly, along with an associated probability for each segmentation. Thus, it is possible, for example, to enumerate the most probable segmentations, which provides more information than the typical approach, in which only a single segmentation is determined. A second advantage to the approach is that no thresholds or arbitrary stopping criteria are needed. Furthermore, no parameter estimation is performed during the process of determining highly probable segmentations. Thus, the method avoids the problems encountered by parameter estimation schemes when there are outliers in the data or when data sets are small, including the accumulation of errors that can occur when estimated parameters are used in subsequent grouping steps.

In this paper we extend the probabilistic framework proposed by LaValle to textured images, and we refine much of the notation proposed by LaValle and Castaño. In Section 2 we present the Markov random field texture model, which will be used for the remainder of the paper. We then describe how we model the difference between our predictions and the input textures. Given the model for texture, in Section 3 we derive the probability that two disjoint regions are homogeneous. In Section 4 we combine the texture model from Section 2 and the probability of homogeneity expression from Section 3, with the result being a probability distribution on the space of possible segments in the image. Precise algorithms are presented later, in Section 6. The framework for segments developed in Section 4 is extended in Section 5 to consider segmentations of an image. Two algorithms are presented in Section 6. First, an algorithm for finding the distribution of segments introduced in Section 4 is presented. Second, an algorithm for finding the distribution of segmentations from Section 5 is presented. In Section 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 discussed in Section 8.

2 Markov Random Fields as Texture Models

An MRF formulation models 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 can be modeled as a linear combination of its four-neighbors, plus Gaussian white noise.

More formally, let an image I be an R x C 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. Then $x_{(r,c)}$ is an observation of that random variable, or the actual intensity value of pixel (r,c) in a given 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 defined as any random field for which the random variable at each pixel is dependent only on the values taken by the random variables for its neighboring pixels in some finite neighborhood which is typically less than the entire image. Formally,

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

$$= P(X_{(r,c)} = x_{(r,c)} | X_{(r',c')} = x_{(r',c')}, (r',c') \in N_{(r,c)}) \tag{1}$$

where $N_{(r,c)}$ is the set of neighbors for pixel (r,c).

The *order* of an MRF defines the size of $N_{(r,c)}$. For example, consider a first order MRF. Pixel (r,c) interacts only with the pixels above, below, to the left, and to the right of itself. Specifically, if we define u_1, u_2, u_3, and u_4 to be the relative weighting of the four-neighbors of (r,c), the linear model for the predicted intensity value of (r,c), defined by a first-order MRF model is

$$\begin{aligned}\hat{x}_{(r,c)} &= u_1(x_{(r-1,c)} - \mu_k) + u_2(x_{(r+1,c)} - \mu_k) + \\ &\quad u_3(x_{(r,c-1)} - \mu_k) + u_4(x_{(r,c+1)} - \mu_k) + \mu_k,\end{aligned} \tag{2}$$

where μ_k represents the mean over region k. We will formally define *region* below. More generally, for an m^{th} order MRF, if we denote the intensity value of the the l^{th} neighbor of (r,c) by $T_l(r,c)$, (2) becomes

$$\hat{x}_{(r,c)} = \left[\sum_{l=1}^{N} u_l(T_l(r,c) - \mu_k)\right] + \mu_k, \tag{3}$$

where u_l again denotes the relative weighting of the l^{th} neighbor. There are N neighbors interacting with pixel (r,c). This formulation was introduced by Kashyap and Chellappa in [17].

If a naturally occurring texture is modeled by a random field, the actual 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. This enables us 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 (r, c) differs from the prediction given in (3) 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.

We define a region, R_k, to be some connected set of pixels. Regions are disjoint from other regions, and the union of all regions in an image is that image. It is useful to discuss vectors of these variables. The vector of random variables associated with the points in a region R_k is denoted $\mathbf{X_k}$. An instance of this random vector is denoted $\mathbf{x_k}$.

The probability density for the observed intensity of a pixel (r, c) is

$$p(x_{(r,c)}|\mathbf{u_k}) = (2\pi\sigma_k^2)^{-\frac{1}{2}} exp\left\{-\frac{1}{2\sigma_k^2}\left[x_{(r,c)} - \hat{x}_{(r,c)}\right]^2\right\} \tag{4}$$

where σ_k^2 is the noise variance over the region R_k, $\mathbf{u_k}$ represents the vector of relative weights $[u_1 u_2 u_3 ... u_N]$ for the neighboring pixels, and $\hat{x}_{(r,c)}$ is as given in (3).

Since we are assuming pixel interactions of a particular form, a proper probability density function would have to model these interactions in all possible combinations. Since we do not do this, the joint density over the points in a region R_k is not a proper pdf. However, it has been shown to be a reasonable approximation in many previous experiments [8], [19], [20], [24], [29]. The *degradation model* defines how the intensity values differ from those predicted by the linear model described in (3). 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 of the N pixels in R_k:

$$p(\mathbf{x_k}|\mathbf{u_k}) = (2\pi\sigma_k^2)^{-\frac{|R_k|}{2}} exp\left\{-\frac{1}{2\sigma_k^2}\sum_{(r,c)\in R_k}\left[x_{(r,c)} - \hat{x}_{(r,c)}\right]^2\right\}. \tag{5}$$

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.

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 [18]. This leads to many problems relating to parameter estimation techniques [13].

We will instead follow LaValle and Hutchinson [22], taking the degradation model from Section 2 and using it to build a probability distribution over possible segments, where a *segment* is some region grouping in the image. Note that we require an initial segmentation of the image, where each region is homogeneous. 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 $p(\mathbf{x_k}|\mathbf{u_k})$, 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, $\mathbf{x_k}$. We will make extensive use of the joint probability density functions (pdfs) of $\mathbf{X_k}$ and $\mathbf{U_k}$ (the random variables of which $\mathbf{x_k}$ and $\mathbf{u_k}$ are instances), which we denote $p(\mathbf{x_k})$ and $p(\mathbf{u_k})$, respectively. Let $p(\mathbf{x_k}, \mathbf{u_k})$ be the combined joint pdf. The joint pdf can be represented as

$$p(\mathbf{x_k}, \mathbf{u_k}) = p(\mathbf{x_k}|\mathbf{u_k})p(\mathbf{u_k}). \tag{6}$$

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

$$p(\mathbf{x_k}) = \int p(\mathbf{x_k}, \mathbf{u_k})d\mathbf{u_k} = \int p(\mathbf{x_k}|\mathbf{u_k})p(\mathbf{u_k})d\mathbf{u_k}. \tag{7}$$

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 $\mathbf{u_i} = \mathbf{u_j}$. For notational convenience, we will use H to represent the condition $H(R_i \cup R_j) = true$.

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

$$\begin{aligned} P(H|\mathbf{x_i}, \mathbf{x_j}) &= \frac{p(\mathbf{x_i}, \mathbf{x_j}|H)P(H)}{p(\mathbf{x_i}, \mathbf{x_j})} \\ &= \frac{p(\mathbf{x_i}, \mathbf{x_j}|H)P(H)}{p(\mathbf{x_i}, \mathbf{x_j}|H)P(H) + p(\mathbf{x_i}, \mathbf{x_j}|\neg H)P(\neg H)}. \end{aligned} \tag{8}$$

The denominator of (8) 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 (8) into λ_r, those terms dealing with regions R_i and R_j, and λ_p, those dealing with priors. Specifically, if we define

$$\lambda_p = \frac{1 - P(H)}{P(H)}, \text{ and } \lambda_r(\mathbf{x_i}, \mathbf{x_j}) = \frac{p(\mathbf{x_i}, \mathbf{x_j} | \neg H)}{p(\mathbf{x_i}, \mathbf{x_j} | H)}, \tag{9}$$

then we can write (8) as

$$P(H | \mathbf{x_i}, \mathbf{x_j}) = \frac{1}{1 + \lambda_p \lambda_r(\mathbf{x_i}, \mathbf{x_j})}. \tag{10}$$

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 $p(\mathbf{x_i}, \mathbf{x_j} | \neg H)$. Our assumption, formally stated as

$$p(\mathbf{x_i}, \mathbf{x_j} | \neg H) = p(\mathbf{x_i}) p(\mathbf{x_j}), \tag{11}$$

allows us to write $\lambda_r(\mathbf{x_i}, \mathbf{x_j})$ as

$$\lambda_r(\mathbf{x_i}, \mathbf{x_j}) = \frac{p(\mathbf{x_i}) p(\mathbf{x_j})}{p(\mathbf{x_i}, \mathbf{x_j} | H)}. \tag{12}$$

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

$$p(\mathbf{x_i}, \mathbf{x_j} | H) = \int p(\mathbf{x_i}, \mathbf{x_j} | \mathbf{u_{ij}}) p(\mathbf{u_{ij}}) d\mathbf{u_{ij}}. \tag{13}$$

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

$$p(\mathbf{x_i}, \mathbf{x_j} | \mathbf{u_{ij}}) = p(\mathbf{x_i} | \mathbf{u_{ij}}) p(\mathbf{x_j} | \mathbf{u_{ij}}), \tag{14}$$

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 [22]. Combining these results gives a *Bayes' factor*

$$\lambda_r(\mathbf{x_i}, \mathbf{x_j}) = \frac{\left[\int p(\mathbf{x_i} | \mathbf{u_i}) p(\mathbf{u_i}) d\mathbf{u_i}\right] \left[\int p(\mathbf{x_j} | \mathbf{u_j}) p(\mathbf{u_j}) d\mathbf{u_j}\right]}{\int p(\mathbf{x_i} | \mathbf{u_{ij}}) p(\mathbf{x_j} | \mathbf{u_{ij}}) p(\mathbf{u_{ij}}) d\mathbf{u_{ij}}}. \tag{15}$$

Bayes' factors have been used to model selection between nested linear parametric models [30], to model selection between parametric image models [21], and for evidence evaluation in forensic science [1], [5].

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 [12]. 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 paper.

An expansion of (15) 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

$$\lambda_r(\mathbf{y_i}, \mathbf{y_j}) = \frac{\left[\sum_{c=1}^{C} p(\mathbf{y_i}|\mathbf{u_c})p(\mathbf{u_c})\right]\left[\sum_{c=1}^{C} p(\mathbf{y_j}|\mathbf{u_c})p(\mathbf{u_c})\right]}{\sum_{c=1}^{C} p(\mathbf{y_i}, \mathbf{y_j}|\mathbf{u_c})p(\mathbf{u_c})}. \tag{16}$$

There is one class representative per class in (16). 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 [12]. A simpler approach is taken in this paper. We take one homogeneous sample from each class and compute the best estimate $\tilde{u}_c$ using standard singular value decomposition techniques [26]. The resultant set of $\tilde{u}_c$ is then used as the set of class representatives.

4 Segment Classes

In this section, 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 of *Segment Classes* and *Segmentation Classes*, presented in this Section and Section 5, which was proposed in [19], does not depend on which adjacency relationship is used. This paper uses a four-neighbor adjacency relationship, but other relationships could be used.

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 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 1, there are four regions, R_1 through R_4. There are 14 possible segments shown in Table 1.

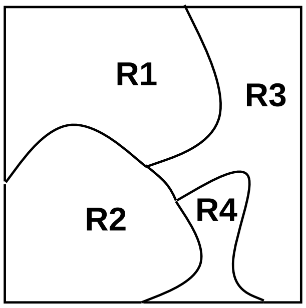


Figure 1: A simple example for describing segments.

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.

Table 1: All possible segments from Figure 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}	$\{R_1 \cup R_2 \cup R_3 \cup R_4\}$

For example, the most general segment class for Figure 1 is $seg(\emptyset, \emptyset)$. A listing of the segments in this class is given in Table 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 be 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 to uncover P_i and find the most probable elements in $seg(\{R_i\}, \emptyset)$.

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

$$P\left[seg(\{R_i, R_j\}, \emptyset | x_k \forall k)\right] = P(H_{ij} | x_k \forall k) \tag{17}$$

where $P(H_{ij})$ is the probability that regions R_i and R_j are homogeneous. It has been shown [4] that a reasonable approximation to (17) using only information from regions R_i and R_j is

$$P\left[seg(\{R_i, R_j\}, \emptyset) | x_k \forall k\right] \approx P\left[seg(\{R_i, R_j\}, \emptyset) | \mathbf{x_i}, \mathbf{x_j}\right] = P(H_{ij} | \mathbf{x_i}, \mathbf{x_j}) \tag{18}$$

Table 2: All possible segment classes from Figure 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}

which yields

$$P\left[seg(\{R_i\}, \{R_j\})|\mathbf{x_i}, \mathbf{x_j}\right] = P(\neg H_{ij}|\mathbf{x_i}, \mathbf{x_j}) = 1 - P(H_{ij}|\mathbf{x_i}, \mathbf{x_j}), \tag{19}$$

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 2 shows three refinements of $seg(\{R_1\}, \emptyset)$, under some $P(H_{12}|\mathbf{x_1}, \mathbf{x_2})$, 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 (d)), which has probability $(0.6)(0.58)=0.25$, and $seg(\{R_1,R_2,R_4\},\{R_3\})$ (lower left bottom of (d)), which has probability $(0.6)(1-0.58)=0.35$.

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 [4]. 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, for which huge classes of segments need not be explicitly considered because of their low probability. Experimentally, we have seen that the number of segment classes considered is typically between 2-4 times the number of regions in the image. In comparison, a (very) loose bound on the number of possible segments would be exponential. In Section 6.1 we will formalize the process of refinement.

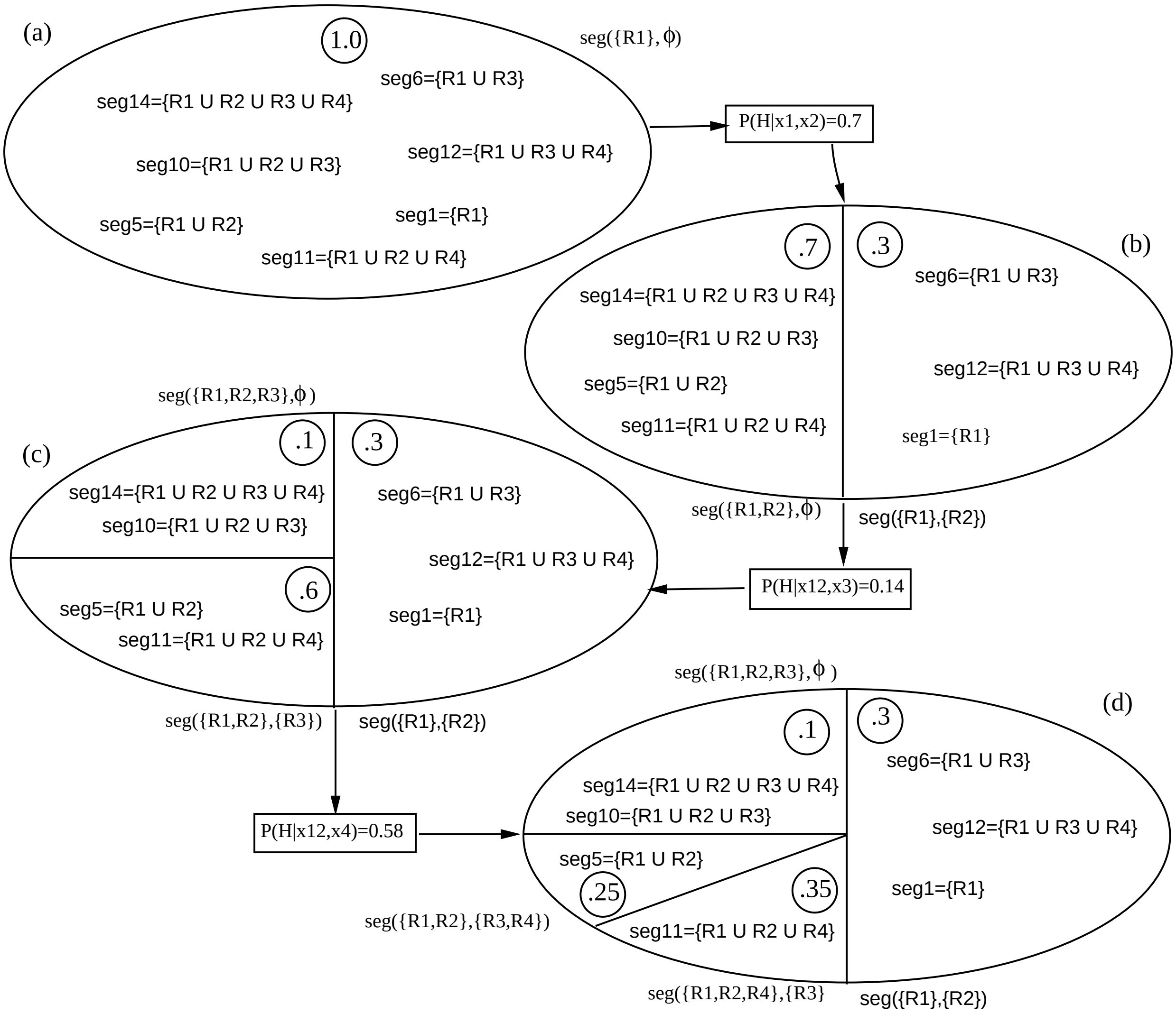


Figure 2: The first four covers of $seg(\{R_1\}, \emptyset)$, (a) $seg(\{R_1\}, \emptyset)$. (a) is refined into (b), (b) into (c), and (c) into (d).

5 Segmentation Classes

In this section, 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 Section 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 Section 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 1 will help to clarify this procedure. First, we use the procedure described in Section 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 3), and we have a ground segmentation class. If the residue is not empty, the

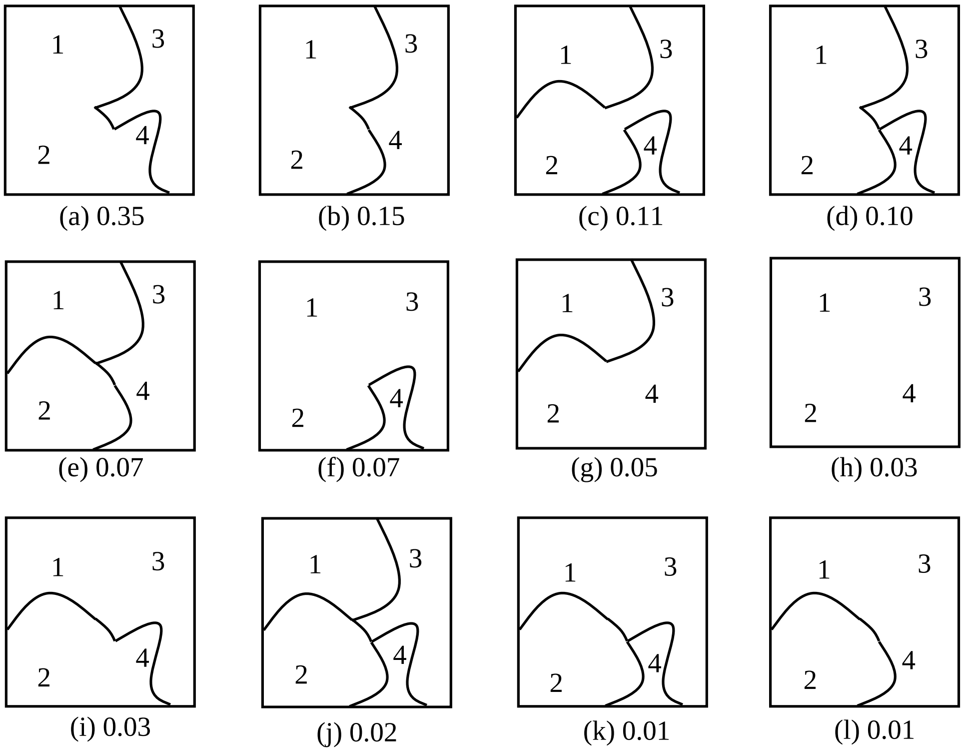


Figure 3: Ground segmentation classes for our simple example in decreasing order of probability.

procedure is used again to obtain all possible segments that make up the residue. If there is only one region left in the residue ((a),(f),(l) in Figure 3), that region will be returned as a segment. 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 Section 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 3. In section 4.2 we derived the probabilities that certain segments are in the correct segmentation of the image I. We saw that 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 (upper left bottom of (d) in Figure 2). Because this is a hypothetical 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, we also saw that 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, 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 4. In the next section, we

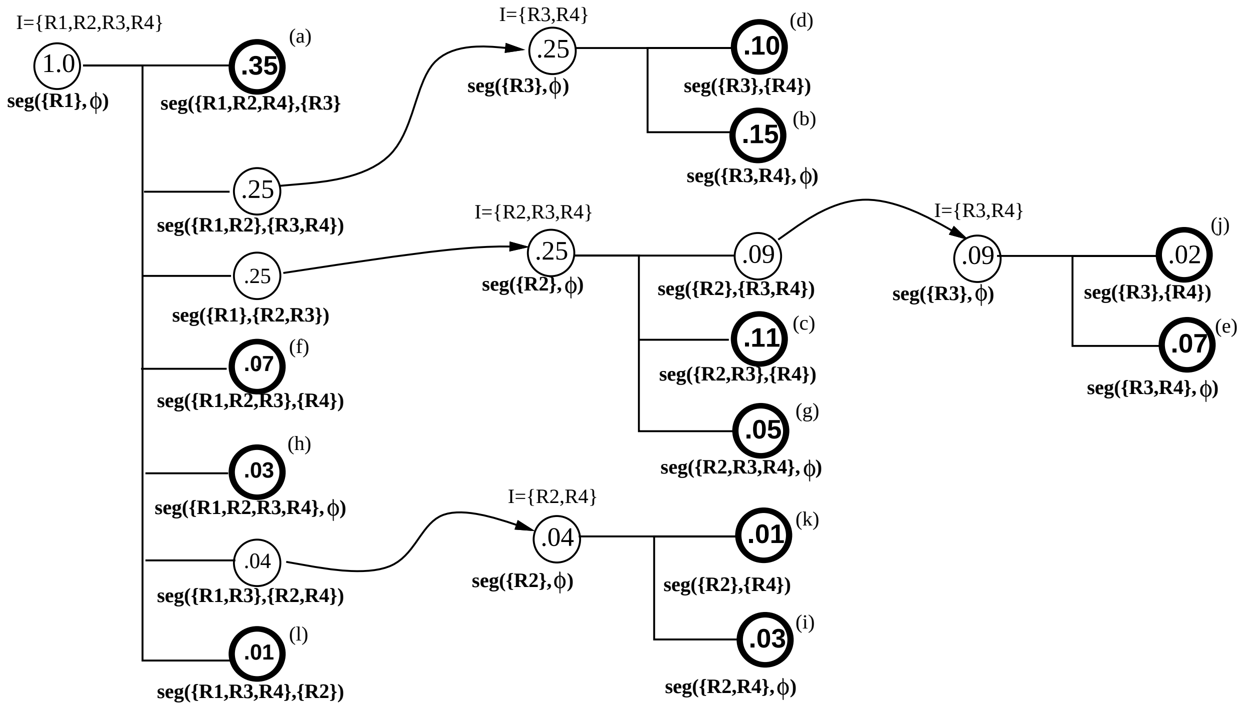


Figure 4: Segmentation classes for our simple example. Ground segmentations are in bold.

shall formalize procedures for obtaining the n most probable segmentations of an image.

This computational savings of a beam search over an exhaustive search can be significant in large images, for which classes of segmentations need not be explicitly considered because of their low probability. Experimentally, we have seen that the number of segmentation classes considered is typically between 2-4 times the number of regions in the image. In comparison, a (very) loose bound on the number of possible segments would be the Bell number, which is super-exponential in the number of regions in the image.

6 Algorithm Issues

In Sections 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 5 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. ExtractMax simply extracts the segment class with the highest probability from the priority queue.

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

$segQ \leftarrow seg(\{R_i\},\{\}); segQ_g \leftarrow \emptyset$; $k \leftarrow 0$
repeat
    $k \leftarrow k+1$
    $seg_k(I_k, E_k) \leftarrow$ ExtractMax(seg)
    select R_k from $\mathcal{R} - I_k - E_k$
    $seg_{k_I} \leftarrow seg(I_k \cup \{R_k\}, E_k)$
    $seg_{k_E} \leftarrow seg(I_k, E_k \cup \{R_k\})$
    calculate probabilities for seg_{k_I} and seg_{k_E}
    if seg_{k_I} is a ground segment class
        INSERT$(segQ_g, seg_{k_I})$
        INSERT$(segQ_g, seg_{k_E})$
    else
        INSERT$(segQ, seg_{k_I})$
        INSERT$(segQ, seg_{k_E})$
until $segQ = \emptyset$ OR
        $(|segQ_g| \geq n$ AND $P(n^{th}\ element\ of\ segQ_g) > P(max\ element\ of\ segQ))$
if $(|segQ_g| > n)$ truncate $segQ_g$ to n elements
return $segQ_g$

Figure 5: 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, 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 $\overline{S}$ 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.

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

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

Figure 6: An algorithm returning the n most probable segmentations.

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

$\mathcal{SEGQ}_g \leftarrow \emptyset; \mathcal{SEGQ} \leftarrow \emptyset$
select initial R_i
INSERT$(\mathcal{SEGQ}, SEG\,[\emptyset, \{R_i\}, \emptyset])$
repeat
  $SEG\,[S_k, \emptyset, \emptyset] \leftarrow$ EXTRACTMAX$(\mathcal{SEGQ})$
  select new R_i from $\mathcal{R} - \overline{S}_k$
  $S_g \leftarrow$ GET-MOST-PROBABLE-SEGMENTS$(\mathcal{R} - \overline{S}_k, R_i, b)$
  for $seg(I_g, E_g) \in S_g$ **do**
    if $(\mathcal{R} - \overline{S}_k - I_g) = \emptyset$ [Partition]
      then
        INSERT$(\mathcal{SEGQ}_g, SEG\,[S_k \cup I_g, \emptyset, \emptyset])$
      else
        INSERT$(\mathcal{SEGQ}, SEG\,[S_k \cup I_g, \emptyset, \emptyset])$
until $\mathcal{SEGQ} = \emptyset$ OR
  $(|\mathcal{SEGQ}_g| \geq n$ AND $P(n^{th}\ element\ of\ \mathcal{SEGQ}_g) > P(max\ element\ of\ \mathcal{SEGQ}))$
if $(|\mathcal{SEGQ}_g| > n)$ truncate $\mathcal{SEGQ}_g$ to n elements
return $\mathcal{SEGQ}_g$

Figure 7: An algorithm that performs beam search on the space of segmentations.

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.

6.4 Initial Region Selection

It should be noted that the initial region from which to expand segment classes (R_i in Figure 5 and Line 15 in Figure 6) must be chosen from a group of appropriate regions. We use a heuristic that selects an initial region by evaluating the maximum conditional probability of the available regions with respect to any model in the model database. The idea is that if a region fits any one of the models in the database well, it could be a good one to expand from. Emperically, the results do not seem to be very sensitive to the heuristic used.

7 Experimental Analysis and Illustrations

The algorithms presented in Section 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 [3] (Figures 8 and 12), from a book on Quilts [14] (Figures 9 and 13), and from [25] (Figures 10, 11, and 14). Distributions of segments and distributions of segmentations were generated for several test images.

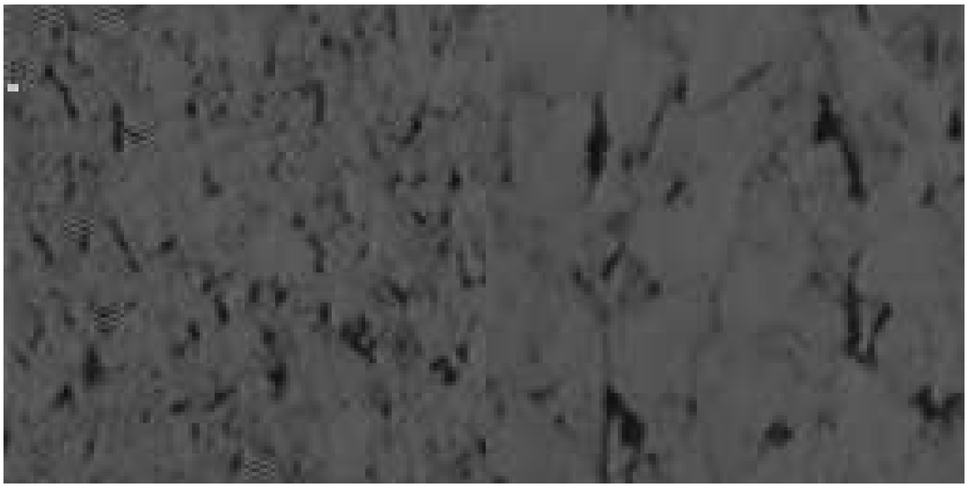
Since third-order MRFs have a small neighborhood, low-frequency intensity variations will not be detected using this model for texture discrimination. It has been noted that even on the homogeneous, uniformly lit images from the Brodatz album, small subimage sizes cause misclassification of some textures [27]. 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. This accounts for the relatively large region size in the following experiments.

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 23 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 (3).

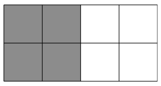
7.1 Segment Distributions

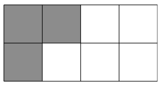
This section presents distributions of segments for some images. In each case, an initial region was chosen, and a distribution of segments was returned. The algorithm shown in Figure 5 was tested on Brodatz textures (Figure 8), unfamiliar textures (Figure 9, and natural scenes (Figures 10 and 11).

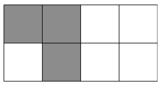
As described above, a selected group of textures from the Brodatz database were used in the generation of class representatives used in the experiments. The first category of experiments, dealt with texture mosaics made from these textures. Figure 8 shows a representative segment distribution produced from one image from this set. This situation, predictably, gives the best results. Each result is labeled with the size of the set of class representatives used in the experiment. For example, the second set was used in Figure 8 so the caption reads "There are 94 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.

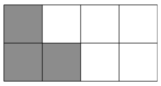


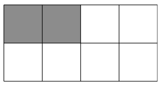
Input Image

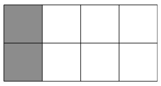

1: 9.97718e-1

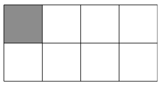

2: 2.26039e-3

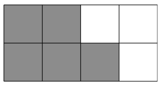

3: 1.06246e-5

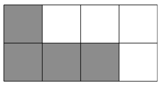

4: 1.06246e-5

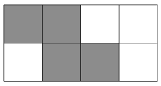

5: 2.40708e-8

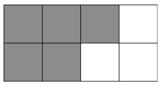

6: 2.40708e-8

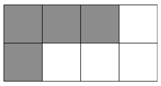

7: 1.13398e-10

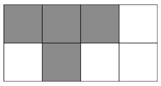

8: 2.18524e-60

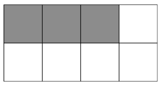

9: 1.44565e-64

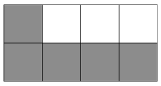

10: 2.9093e-65

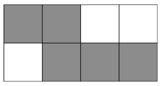

11: 2.24444e-67

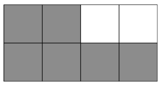

12: 5.08492e-70

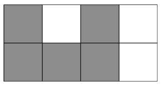

13: 1.29875e-70

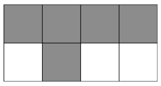

14: 1.32587e-73


15: 2.89613e-95


16: 1.39023e-96


17: 5.69512e-103


18: 1.04148e-107


19: 8.05014e-109

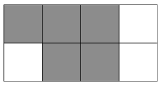
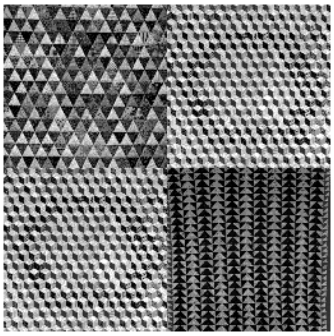

20: 4.99945e-109

Figure 8: The 20 top segments. The region size is 64. There are 94 models. This image was used in unsupervised training of the algorithm.

The second category of experiments dealt with texture mosaics from [14] that were similar to the images used above. 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 9 shows a representative segment distribution produced from one image from this set.



Input Image

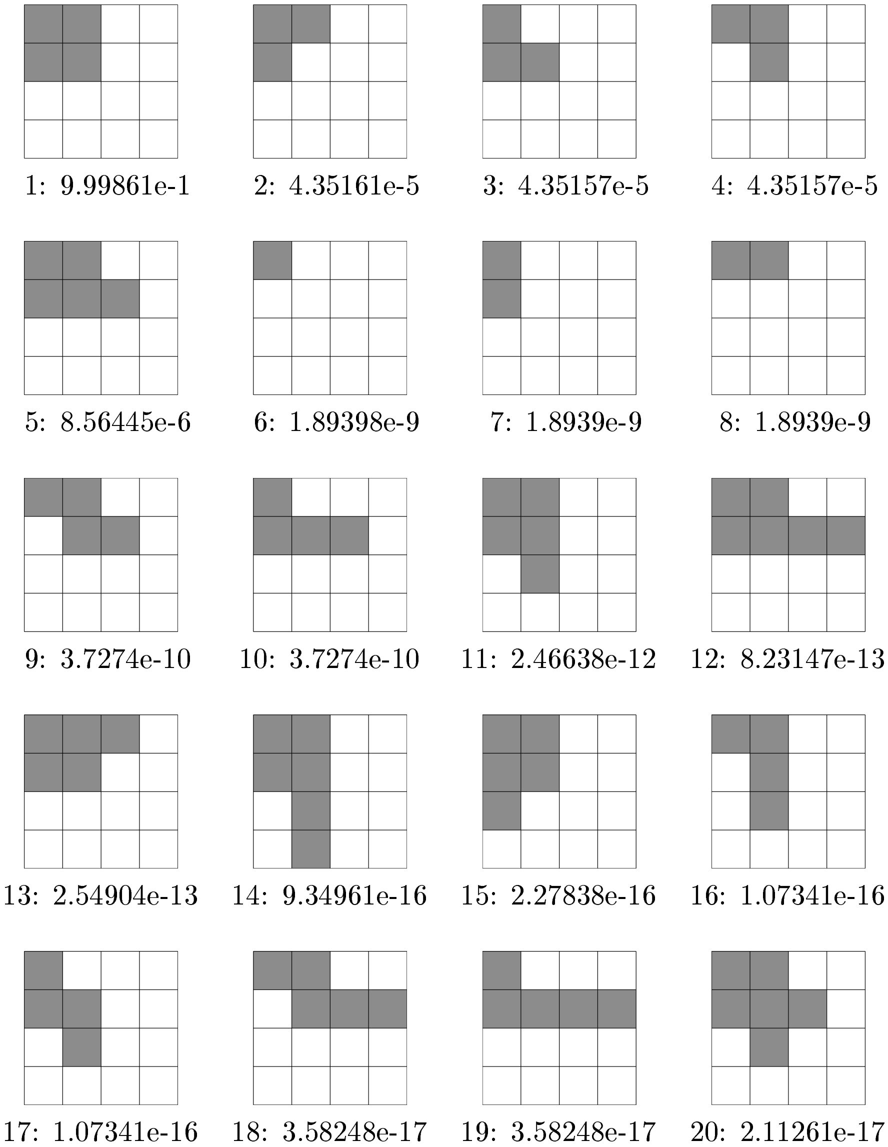
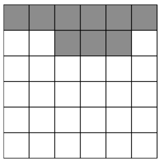


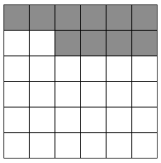
Figure 9: The 20 top segments. The region size is 64. There are 23 models. This image has not been previously seen by the algorithm.

The final set of experiments involving distributions of segments examines the algorithm's performance on more natural scenes from [25]. Not surprisingly, the performance on these images, shown in Figures 10 and 11, 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 trees in the upper portion of Figure 10.

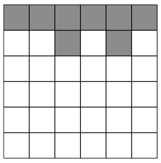
Input Image



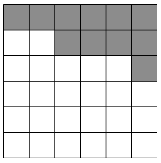
1: 9.01903e-1



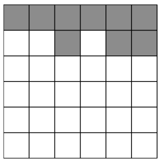
2: 7.81645e-2



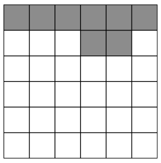
3: 1.25576e-2



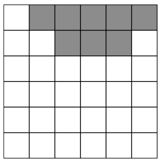
4: 4.98652e-3



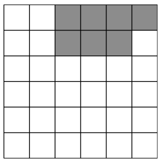
5: 1.17463e-3



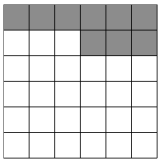
6: 7.31345e-4



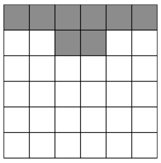
7: 1.27141e-4



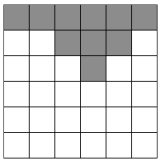
8: 6.56752e-5



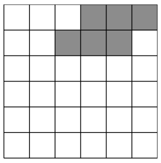
9: 6.47763e-5



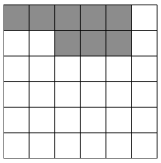
10: 6.01086e-5



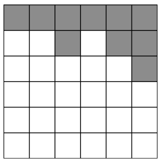
11: 5.15713e-5



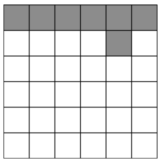
12: 1.99879e-5



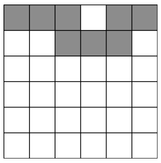
13: 1.69273e-5



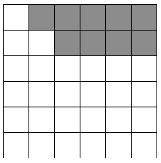
14: 1.24927e-5



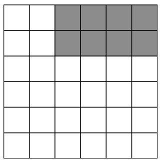
15: 1.21117e-5



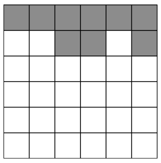
16: 1.18585e-5



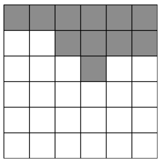
17: 9.84289e-6



18: 5.15613e-6



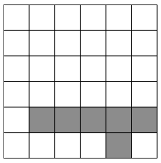
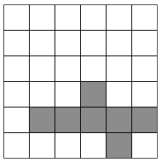
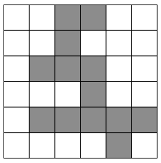
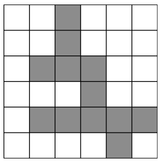
19: 4.68246e-6



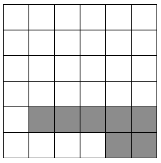
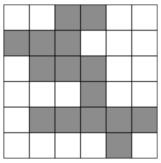
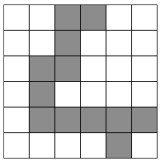
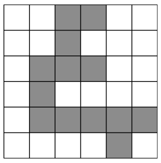
20: 4.57626e-6

Figure 10: The 20 top segments. The region size is 32. There are 94 models. This image has not been previously seen by the algorithm.

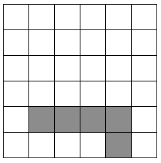
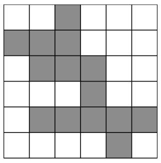
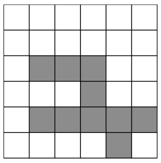
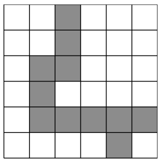
Input Image

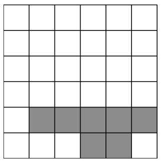
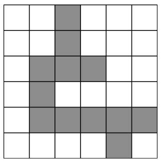
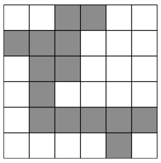
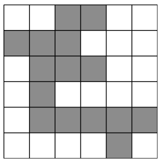
1: 9.66383e-1 2: 1.75205e-2 3: 7.84116e-3 4: 2.42226e-3

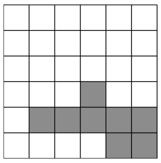
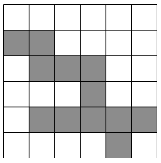
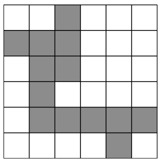
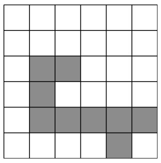
5: 2.15682e-3 6: 9.74691e-4 7: 5.80703e-4 8: 3.91974e-4

9: 3.16607e-4 10: 3.01097e-4 11: 2.47011e-4 12: 1.79388e-4

13: 1.48532e-4 14: 1.21087e-4 15: 7.21839e-5 16: 4.87241e-5

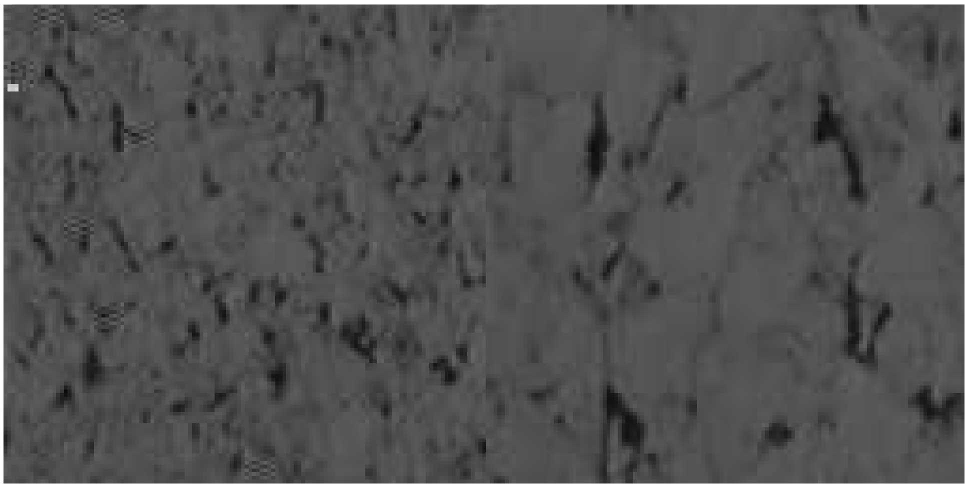
17: 3.9103e-5 18: 3.07045e-5 19: 2.22987e-5 20: 1.82932e-5

Figure 11: The 20 top segments. The region size is 32. There are 94 models. The initial region was Region 28 (near the center of the water.) This image has not been previously seen by the algorithm.

7.2 Segmentation Distributions

Now we turn to full segmentations and compute distributions of segmentations for the same types of images used above. The algorithm shown in Figure 7 was tested on Brodatz textures (Figure 12), unfamiliar textures (Figure 13, and natural scenes (Figure 14).

The texture mosaics used in this set of experiments were the same ones used in Section 7.1, from [3]. 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 12 shows a representative segmentation distribution produced from one image in this set.



Input Image

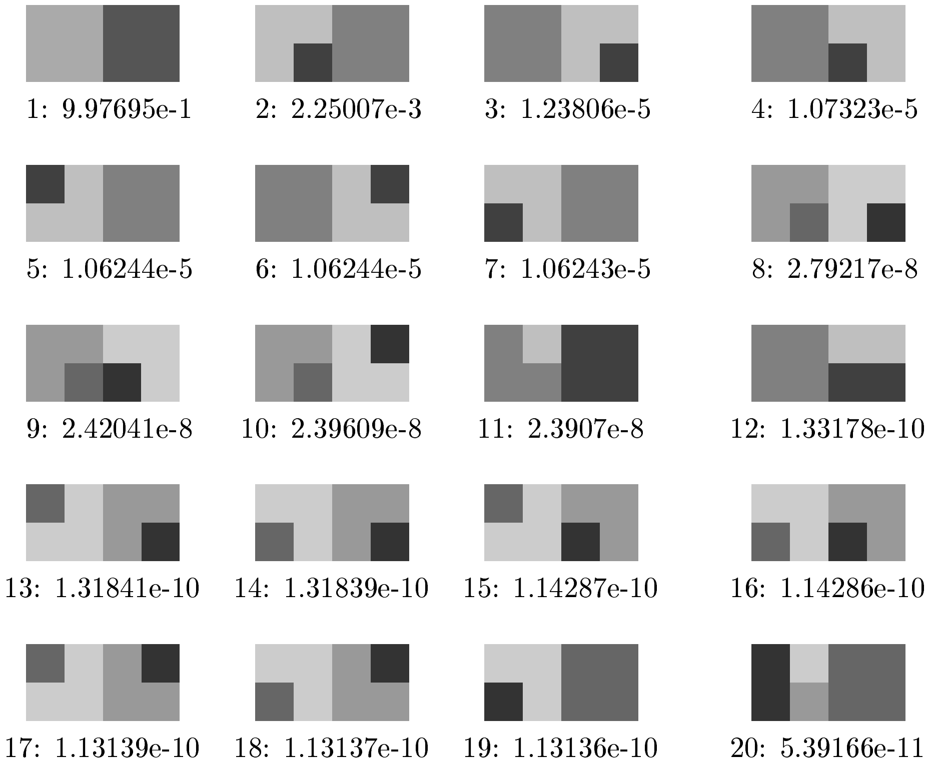


Figure 12: The 20 top segmentations. The region size is 64. There are 94 models. This image was used in unsupervised training of the algorithm.

As in the second class of experiments above, these mostly homogeneous high-frequency textures from [14] were not in any way used in the generation of class representatives for the algorithm. Figure 13 shows a representative segmentation distribution produced from one image from this set.

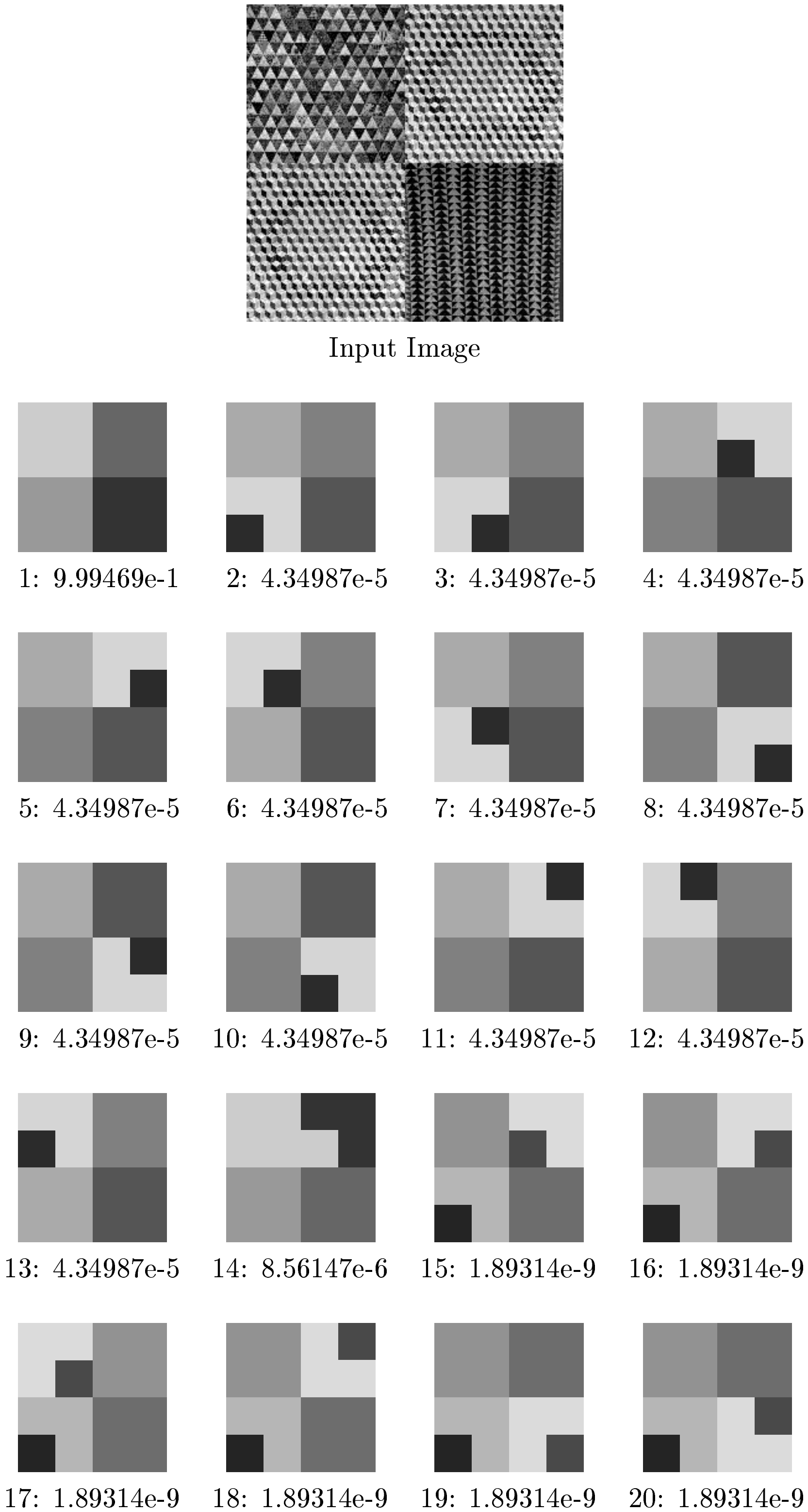
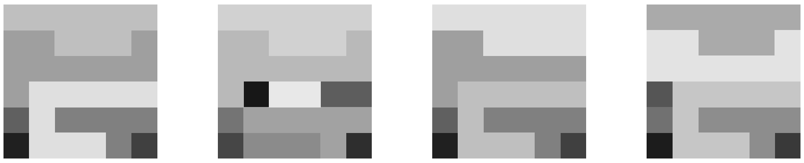


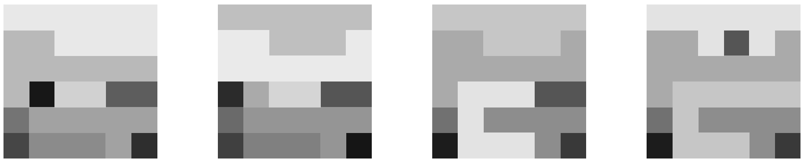
Figure 13: The 20 top segmentations. The region size is 64. There are 23 models. This image has not been previously seen by the algorithm.

The final set of experiments involving distributions of segmentations examines the algorithm's performance on more natural scenes from [25]. Figure 14 shows that while the performance is not as good as in the previous experiments, that the algorithm does tend to group regions containing homogeneous, high-frequency textures. For example, the trees in Figure 14 are grouped, since the leaves have what could be considered a homogeneous texture.

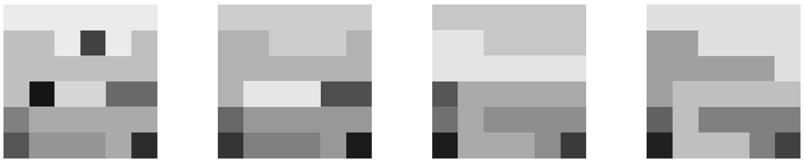
Input Image



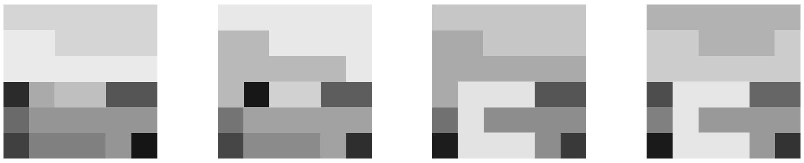
1: 5.09936e-1 2: 2.99146e-1 3: 4.42096e-2 4: 4.3093e-2



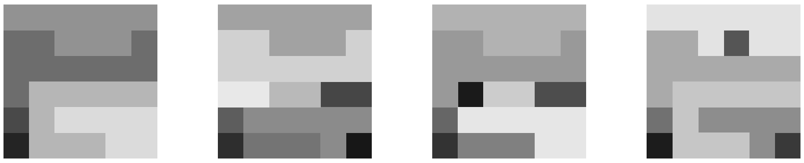
5: 2.59349e-2 6: 2.4216e-2 7: 1.36558e-2 8: 7.10007e-3



9: 4.16515e-3 10: 4.07786e-3 11: 3.73601e-3 12: 2.82039e-3



13: 2.09944e-3 14: 1.65454e-3 15: 1.18391e-3 16: 1.15401e-3



17: 1.1381e-3 18: 1.06394e-3 19: 6.67649e-4 20: 6.64366e-4

Figure 14: The 20 top segmentations. The region size is 32. There are 94 models. This image has not been previously seen by the algorithm.

8 Conclusions and Future Research

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. Extensive work has been done in this field, but the results of these models on natural textures show that there is room for improvement [6], [9], [10].

Also of interest is how domain knowledge could influence the prior distribution in the Bayesian analysis. To go back to the example of Section 3, if a facet model is in use and it is known that all planes meet at right angles, (11) 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 [19] 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 [22], [31], and represent promising directions for the integration of low-level and mid-level computer vision algorithms.

REFERENCES

[1] 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.

[2] 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.

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

[4] R. L. Castaño and S. A. Hutchinson. A framework for probabilistically evaluating image feature groupings. *Comput. Vis. Image Understanding*, Accepted for publication.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

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

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

[11] 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.

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

[13] 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.

[14] J. Holstein. *American Pieced Quilts.* The Viking Press, New York, 1972.

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

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

[17] 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.

[18] S. Lakshmanan and H. Derin. Simultaneous parameter estimation and segmentation of Gibbs random fields using simulated annealing. *IEEE Trans. Pattern Anal. Mach. Intell.*, 11(8):799–813, August 1989.

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

[20] 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.

[21] 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.

[22] 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.

[23] 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.

[24] 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.

[25] D. K. Panjwani and G. Healey. Markov random field models for unsupervised segmentation of textured color images. *IEEE Trans. Pattern Anal. Mach. Intell.*, (10):939–954, October 1995.

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

[27] 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.

[28] 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.

[29] 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.

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

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