Multiple Resolution Segmentation of Textured Images [†]

Charles Bouman
School of Electrical Engineering
Purdue University
West Lafayette, IN 47907
(317) 494-0340

Bede Liu
Department of Electrical Engineering
Princeton University
Princeton, New Jersey 08544
(609) 258-4628

Abstract

This paper presents a multiple resolution algorithm for segmenting images into regions with differing statistical behavior. In addition, an algorithm is developed for determining the number of statistically distinct regions in an image and estimating the parameters of those regions. Both algorithms use a causal Gaussian autoregressive (AR) model to describe the mean, variance and spatial correlation of the image textures. Together the algorithms may be used to perform unsupervised texture segmentation.

The multiple resolution segmentation algorithm first segments images at coarse resolution and then progresses to finer resolutions until individual pixels are classified. This method results in accurate segmentations and requires significantly less computation than some previously known methods. The field containing the classification of each pixel in the image is modeled as a Markov random field (MRF). Segmentation at each resolution is then performed by maximizing the *a posteriori* probability of this field subject to the resolution constraint. At each resolution, the *a posteriori* probability is maximized by a deterministic greedy algorithm which iteratively chooses the classification of individual pixels or pixel blocks.

The unsupervised parameter estimation algorithm determines both the number of textures and their parameters by minimizing a global criteria based on the AIC information criterion. Clusters

[†]This work was supported by the Office of Naval Research and the Naval Research Laboratory under Grant N00014-87-0189 and by a IBM Graduate Fellowship.

corresponding to the individual textures are formed by alternately estimating the cluster parameters and repartitioning the data into those clusters. Concurrently, the number of distinct textures is estimated by combining clusters until a minimum of the criteria is reached.

1 Introduction

The objective of texture segmentation is to separate an image into regions of distinct statistical behavior. An implicit assumption in this process is that the statistics of each region are stationary and that each region extends over a significant area. Therefore, it is reasonable to assume that image pixels which are spatially close are likely to be of the same texture. In addition, any texture segmentation algorithm which independently classifies each pixel in an image is likely to perform poorly since locally there may not be sufficient information to make a good decision. For these reasons, most segmentation algorithms either implicitly or explicitly impose some form of smoothness in the resulting segmentation. For example, this may be done by dividing the image into arbitrary blocks and classifying the texture of each block separately. However, if the block size chosen is too small, discriminating among similar textures may be difficult. Alternatively, if the block size is too large, regions of differing texture may be lost. In either case, the resulting boundaries will not be accurate since there is no reason to believe that the actual texture boundaries occurred along the block boundaries. Alternatively, a number of authors have proposed more natural methods of imposing smoothness constraints on the segmentation of an image [6, 4, 3, 2, 1, 11]. These methods use a random field with smooth spatial behavior to model the discrete valued field containing the classification of each pixel in the image. Segmentation is then approached as a statistical estimation problem. In this approach, a region in the image is classified differently from the surrounding regions if there is sufficient statistical evidence to justify a distinct region regardless of size.

These methods of segmentation use Markov random fields (MRF) to model the discrete field containing the individual pixel classifications. In an MRF, each point is statistically dependent only on its neighbors so that the complexity of the model is restricted. The image is then segmented by finding an approximate maximum *a posteriori* (MAP) estimate of the unknown field since this is the most likely segmentation given the observed data. However, the resulting functions are quite difficult to maximize globally. Stochastic relaxation algorithms such as simulated annealing

(SA) have been applied to these problems [6, 2]. It has been shown that, with the proper annealing schedule, SA will converge to the global optimum [6], but such a schedule converges much too slowly to be practically useful. In practice, an annealing schedule is chosen to yield a reasonable trade off between performance and computation, but SA remains a computationally intensive method of minimization.

Alternatively, a greedy minimization algorithm known as estimation by iterated conditional modes (ICM) [4] is computationally efficient. This method maximizes the probability of the segmentation field by deterministically and iteratively changing pixel classifications. When textures can be discriminated over small regions containing few pixels, ICM yields good results. However, our work indicates that when larger numbers of pixels are necessary to discriminate among different textures, as is likely in higher resolution images, ICM is prone to being trapped in local minimum of the cost function.

The algorithm developed in this paper first segments the image at coarse resolution and proceeds to progressively finer resolutions until individual pixels are classified. At each resolution a greedy algorithm is used to perform the classification, and then the result is used as an initial condition at the next finer resolution. It is found that this multiple resolution segmentation (MRS) algorithm is less likely to be trapped in local minima than ICM since at each resolution the regions of appropriate size are classified and used to guide finer resolutions. This is consistent with the observation that the convergence of MRF based segmentation algorithms can be improved by varying the local neighborhoods [3]. In addition, the MRS algorithm requires substantially less computation than ICM since constraints can propagate rapidly across the image at coarse resolution. These advantages can be important in high resolution images where individual pixels contain relatively little information, and in the discrimination between textures where information may be spread over large regions.

The approach developed in this paper is analogous to the multigrid methods used in computing solutions to partial differential equations. Multigrid methods, which have recently been applied to some computer vision problems [13], substantially reduce computation since constraints propagate rapidly at coarse resolution. However, when solving linear partial differential equations, multigrid methods do not improve the quality of the solution since local algorithms are guaranteed to converge to the unique solution (if finite precision effects are ignored). In contrast, multiple resolution segmentation techniques can both reduce computation and improve performance since local minima in the cost function may be avoided by allowing coarse resolution solutions to guide finer solutions [11, 12].

In order to use algorithm in an unsupervised fashion, a method is needed to estimate the number of texture regions and their parameters. Our approach is to first develop a statistical model of the observed image based on the texture model used in segmentation. This overall model is parameterized by the number of textures, the parameters of the individual textures, and the rate at which each texture occurs. It is then shown that, for a fixed number of textures, approximate MLE parameter estimation may be performed by alternately reestimating the texture parameters and repartioning the clusters associated with each texture. This operation is analogous to algorithms such as the K-means algorithm [17] but maximizes the log likelihood instead of minimizing the mean square deviation about cluster means.

The problem of estimating the number of textures is quite different from estimating their parameters since the MLE of the number of textures may be arbitrarily large. It has been suggested [15, 16] that a reasonable alternative to the MLE is to find the number and type of regions which minimize an information criteria (AIC) proposed by Akaike [14]. The AIC contains an additional term to account for the bias due to the number of parameters to be estimated. We adopt a similar approach, but based on a modified criteria. We also show that this criteria may be minimized through the selective agglomeration of texture clusters and the application of the parameter estimation algorithm described above. In this scheme, the decision to agglomerate two clusters is based on a cost function which equals the change in the criteria when nodes are merged. This cost function is shown to be a function of the texture statistics associated with each of the two clusters. As a final step, segmentation is performed at coarse resolution, and data occurring along boundaries are removed so that superfluous textures formed at boundaries can be eliminated.

Both the segmentation and parameter estimation algorithms use a nonhomogeneous Gaussian autoregressive (AR) model for the individual textures. The model allows the extraction of a texture statistic for each pixel which can be averaged over arbitrary regions of pixels to yield an aggregate statistic for the region, and is therefore well adapted to use with multiple resolution techniques.

The advantages of a predictive texture model over a model which uses only mean and variance is shown to be particularly important in estimating the number of distinct textures when the textures have high spatial correlation.

2 Statistical Model

Throughout this paper, we will assume that the observed image, Y, is a random field defined on a rectangular grid, S, of N points, and the value of Y at a point $s \in S$ will be written as Y_s. When necessary the points in S will be explicitly written as integer pairs (i, j). X will denote the field, also of N points, which contains the the classification of each pixel in Y. Points in X will take values in the set $\{1, .., M\}$, where M is the number of textures or classifications. Further, the conditional density function of Y given X, is assumed to exist and to be strictly positive and is denoted by $f(y|x)$, and the probability $P(X = x)$ is written as $p(x)$.

Using this framework, the image may be segmented by estimating the pixel classifications X given the observed image Y. In particular, we will use the MAP estimate of X.

$$\begin{aligned} \hat{x}_{MAP} &= \arg\max_x \{P(X = x|Y = y)\} \\ &= \arg\min_x \{-\log f(y|x) \ - \ \log p(x)\} \qquad (1) \end{aligned}$$

Since the set of possible segmentations is discrete, the maximum or minimum of a well defined function on the set must exist. Hence, once the distributions for $f(y|x)$ and $p(x)$ are defined, the problem of segmenting an image will reduce to that of minimizing a cost function.

2.1 Texture Model

Given a known segmentation, x, the image Y will be modeled as a causal nonhomogeneous Gaussian AR random field [7]. This model has the advantage that the extraction of texture statistics and the estimation of parameter values are direct and computationally efficient. The causal AR model parameterizes the conditional distribution of a point in the field given past values of points in a local neighborhood. Of course, the concept of a past is dependent on a specific ordering of the image pixels (in this work raster ordering is used). A question which arises is whether the artificial introduction of causality to an image will restrict the range of textures which may be modeled accurately.

A partial answer to this question is found by considering other Gaussian random field models such as simultaneous autoregressive (SAR) and Gauss Markov random field (GMRF) [8] which do not require the introduction of a causal ordering. For Gaussian random fields, the relationship among AR, SAR and GMRF models depends on the neighborhoods which are used. The AR model is a special case of the SAR model obtained by constraining the prediction coefficients for future points to be zero. Also, any SAR model may be equivalently described as a GMRF but with a larger finite neighborhood. However, by choosing a sufficiently large neighborhood size, any of these models may be used to approximate arbitrarily closely a Gaussian random field with well behaved spectral density function [7]. Therefore, the major practical difference among these models is the complexity required to achieve a sufficiently accurate model. Because SAR and GMRF models have the disadvantage that maximum likelihood (ML) parameter estimation is difficult, we choose to use the AR model. This difficulty in parameter estimation arises due to the complex dependence of the distributions normalizing constant on the parameters [8] being estimated.

The image Y is assumed to consist of M textures each of which is described by a parameter vector θ_m where m is the index of the particular texture. Each parameter vector, θ_m, contains the mean, μ_m, prediction variance, σ_m^2, and prediction coefficients, a_m, of that texture. The parameter set used at the point $s \in S$ is then θ_{x_s}. We will assume raster ordering of the points in S, and for $(i,j) \in S$ we will define $(i,j) < (k,l)$ to mean $(i = k \ \& \ j < l)$ or $i < k$. Also, for $s, r \in S$, $s - r$ will denote element by element subtraction of the components of s and r.

A basic property of the AR model is that the prediction errors, formed by filtering with the prediction coefficients, are independent. This property can be used to calculate the conditional distribution of Y given x. The forward prediction error at s for a P^{th} order AR model is given by

$$\tilde{Y}_s = (Y_s - \mu_{x_s}) + \sum_{r>0} a_{x_s}(r)\,(Y_{s-r} - \mu_{x_s}) \tag{2}$$

where there are P values of r for which $a_{x_s}(r)$ may be nonzero. The placement of these P values depends on the prediction window used. We will assume the use of a second quadrant predictor window although other models could also be used. If we ignore the effects of the image boundary, the components of the prediction error $\tilde{Y}_s$ are independent with mean zero and variance

$$E\left[\tilde{Y}_s^2\right] = \sigma_{x_s}^2 \ .$$

Given this information, the conditional distribution of $\tilde{Y}$ given x is just that of N independent Gaussian random variables. However, the Jacobian of the transformation from Y to $\tilde{Y}$ is needed to compute the conditional distribution of Y. This can be calculated by arranging the components of Y and $\tilde{Y}$ into vectors with raster ordering. It is then possible to write (2) in the form

$$\tilde{Y} = C + AY$$

where A is a matrix containing the prediction coefficients and C is a constant vector. Also, A is lower triangular, and its diagonal entries are all 1's. This is due to the fact that in raster ordering all points used in prediction will precede the point being predicted. These facts imply that the Jacobian of the transformation from Y to $\tilde{Y}$ is 1, and that the conditional distribution of Y is

$$f(y|x) = \prod_{s\in S} \frac{1}{\sqrt{2\pi\sigma_{x_s}^2}} \exp\left\{-\frac{1}{2}\sum_{s\in S}\frac{\tilde{y}_s^2}{\sigma_{x_s}^2}\right\} \ .$$

This results in a negative conditional log likelihood of

$$-\log f(y|x) = \frac{1}{2}\sum_{s\in S}\left\{\frac{\tilde{y}_s^2}{\sigma_{x_s}^2} \ + \ \log(\sigma_{x_s}^2) \ + \ \log(2\pi)\right\} \ . \tag{3}$$

The log likelihood function (3) has the form of a sum of local functions of y and x. In general, the methods used throughout this paper are applicable to any texture model which has a log likelihood function of the form

$$-\log f(y|x) = \sum_{s\in S} l_s(y|x_s) + c(y) \tag{4}$$

where the functions l_s depend on all of y but only x at the point s, and $c(\cdot)$ is an arbitrary function of y. Specifically, for this texture model $c(y) = 0$ and the local likelihood functions have the form

$$l_s(y|x_s) = \frac{1}{2}\left\{\frac{\tilde{y}_s^2}{\sigma_{x_s}^2} \ + \ \log(\sigma_{x_s}^2) \ + \ \log(2\pi)\right\} \ . \tag{5}$$

2.2 Segmentation Field Model

We use an MRF to model X, due to its isotropic nature and its restriction to local interactions. In order to define an MRF, the concept of neighborhoods must first be defined. The neighborhood of a point $s \in S$ is a set of points $\partial s \subset S$ with the two properties that $\forall s, r \in S \; s \notin \partial s$ and $s \in \partial r \Leftrightarrow r \in \partial s$. The set of ordered pairs $\{(s, \partial s)\}_{s \in S}$ is then known as a neighborhood system. We will use an eight-point neighborhood system in which the neighbors of interior points in S are given by

$$\partial(i,j) \; = \; \{(i+k, j+l) : k = \pm 1, \; l = \pm 1\} \; .$$

The neighbors of boundary points will depend on whether a toroidal or free boundary is specified.

For an MRF, X, the conditional distribution of a point in the field given all other points is only dependent on its neighbors; that is

$$\forall s \in S \;\; p(x_s | x_{S-s}) = p(x_s | x_{\partial s}) \; .$$

A clique, c, is a subset of points in S such that if s and r are two points contained in c then s and r are neighbors. Notice that the set of all cliques is induced by the neighborhood system. For a given neighborhood system, a Gibbs distribution is defined as any distribution, $p(x)$, which can be expressed in the form,

$$p(x) = \frac{1}{z} \exp\left\{ -\frac{1}{T} \sum_{\text{all cliques } c} V^c(x_c) \right\}$$

where the functions V^c are arbitrary functions of the values of x on the clique c, and z is a normalizing constant. The constant T is physically analogous to temperature, and the argument of the exponential

$$U(x) = \sum_{\text{all cliques } c} V^c(x_c)$$

is physically analogous to energy.

The Hammersley-Clifford theorem [5] states that X is an MRF with strictly positive distribution and a neighborhood system $\{(s, \partial s)\}_{s \in S}$ if and only if the distribution of X can be written as a Gibbs distribution with cliques induced by the neighborhood system $\{(s, \partial s)\}_{s \in S}$. Therefore, if $p(x)$ is formulated as a Gibbs distribution, we know that X will have the properties of an MRF. The only cliques which will be used in the energy function for $p(x)$ are pairs of horizontally, vertically and diagonally adjacent points. The functions of these cliques will appear implicitly through the two functions $t_1(X)$ and $t_2(X)$. $t_1(X)$ is the number of horizontally and vertically neighboring points of different value in X, and $t_2(X)$ is the number of diagonally neighboring points of different value in X. The density function for X is then assumed to be of the form

$$p(x) = \frac{1}{z}\exp\{-\beta_1 t_1(x) - \beta_2 t_2(x)\} \tag{6}$$

where β_1 and β_2 are parameters of the model. Since both t_1 and t_2 may be written as the sum of functions of cliques, we know that (6) is the distribution of an MRF with an eight-point neighborhood. A more general model for MRF's with two point cliques results by allowing a different potential for each distinct combination of unlike pixel pairs, and by adding a term for one pixel cliques. It should be noted that the arguments used in this paper may be easily generalized for this case. However, we will assume that prior knowledge is not available about the behavior of individual class regions.

Substituting in $p(x)$ from (6) and $\log f(y|x)$ from (4) into (1) results in the final form of the minimization problem

$$\hat{x}_{MAP} = \arg\min_x U(x|y)$$

where

$$U(x|y) = \sum_{s\in S} l_s(y|x_s) + \beta_1 t_1(x) + \beta_2 t_2(x) . \tag{7}$$

3 Minimization Methods

In general, the problem of minimizing (7) is quite difficult since the cost function is not convex and contains complex local minima. The problem may be formulated as a one dimensional dynamic programming problem by associating a state with the value of x on a entire line, but the resulting algorithm is computationally infeasible. Derin and Elliott have proposed finding an approximate solution by solving the dynamic programming problem along thin strips [1]; however, the computation required for such an algorithm increases dramatically with the number of textures. Also, while the problem may be solved in one pass, this pass is quite complex and sequential in nature. The approaches explored in this paper are iterative, but generally only require the application of simple local operations, and they are well suited to highly parallel implementations.

3.1 SA and ICM Methods

SA is a general purpose algorithm for minimizing a cost function, $C(x)$, as a function of the state x. SA works by generating an irreducible Markov chain of these states. At each step of the chain, the transition distribution is chosen so that the limiting distribution is a Gibbs distribution with temperature T and energy function $C(x)$. The minimum of $C(x)$ is then found by allowing T to drop slowly as the chain is generated. Since for sufficiently small T the lowest energy state of the Gibbs distribution occurs with high probability, it is reasonable to expect that if T drops slowly the samples will be concentrated about this minimum energy state. An advantage of SA is that it only requires that the change in cost be computed between consecutive states in the Markov chain.

There are two distinct methods of generating samples from such a Gibbs distribution, and consequently two distinct methods of performing SA. Both depend on the ergodic properties of the Markov chains formed by iteratively replacing points in x. In the Metropolis algorithm [9] a symmetric transition probability is defined and then transitions are either accepted or rejected with a probability depending on the change in energy. The alternative method, which will be used in this paper, is the Gibbs sampler [6] which replaces points in x with samples from the conditional distribution of a point given the remainder of the field.

$$p_T(x_s|x_{S-s}) = \frac{\exp\{-\frac{1}{T}C(x)\}}{\sum_{k=0}^{M}\exp\{-\frac{1}{T}C(x_s = k, x_{S-s})\}} \tag{8}$$

Notice that when $C(x)$ is the energy function of an MRF then $p_T(x_s|x_{S-s})$ will only depend on $x_{\partial s}$.

For the segmentation problem formulated in the previous section the cost function is given by $C(x) = U(x|y)$. Because the functions $l_s(y|\cdot)$ are only dependent on individual points in x, it is easily verified that $U(\cdot|y)$ is the energy function of a Gibbs distribution with an eight-point neighborhood. Therefore, by applying the Hammersley-Clifford theorem, we know that the conditional distribution of (8) will only be a function of the eight neighbors of x_s. If T is large, transitions will occur almost uniformly over the set of possible values for x_s; but if T is small, the value of x_s corresponding to the most likely conditional probability will be chosen with high probability.

It has been shown [6] that if T is varied according to a schedule, $T(n)$, with the correct starting value and rate of decrease, then as $n \to \infty$ the distribution of $X(n)$ will be uniform over the values of x which minimize the cost function. In practice this annealing schedule is much too slow, so a faster schedule is used which represents a trade-off between performance and computation. When a faster schedule is used the algorithm is not guaranteed to escape from local minima. For example, if a large region has been misclassified, the algorithm will probably never explore the possibility of changing all of the classifications in that region. For this reason, the initial condition to the SA algorithm is often of critical importance.

Besag has proposed the iterated conditional modes (ICM) algorithm [4] as an alternative to SA. ICM may be regarded as SA with the extreme annealing schedule $T(n) = 0$. ICM replaces points in x with a value which maximizes the conditional density function at each point. Since

$$\begin{aligned}\arg\min_{x_s} U(x_s, x_{S-s}|y) &= \arg\max_{x_s} p_T(x_s, x_{S-s}|y) \\ &= \arg\max_{x_s} p_T(x_s|x_{S-s}, y) \ ,\end{aligned}$$

it is clear that this replacement selects a value for x_s which minimizes the cost under the constraint of fixing the remaining values in x. Let $v_1(x_{\partial s}, k)$ be the number of horizontal and vertical neighbors of x_s which are not equal to k, and let $v_2(x_{\partial s}, k)$ be the number of diagonal neighbors of x_s which are not equal to k. The replacement can then be explicitly expressed as

$$\arg\min_{x_s} U(x_s, x_{S-s}|y) = \arg\min_{x_s} \{ l_s(y|x_s) \ + \beta_1 \, v_1(x_{\partial s}, x_s) \ + \beta_2 \, v_2(x_{\partial s}, x_s) \} \ , \qquad (9)$$

The ICM algorithm only differs from SA with $T(n) = 0$ when the argument which minimizes (9) is not unique. In this case, SA will replace x_s with a value chosen with uniform probability from the minimizing values. However, ICM selects the previous value of x_s if it minimizes the cost. Otherwise, ICM chooses pseudorandomly among the minimizing values. Consequently, each time this replacement operation changes the value of a point in x, the cost, $U(x|y)$, is assured to decrease. Besag suggested these replacements be performed for a fixed number of iterations through x; however, our approach is to continue replacing points in x until no additional changes occur. Since $U(\cdot|y)$ is strictly positive with a finite domain, the convergence of the algorithm in a finite number of steps is assured. The minimizing state x which is obtained is a local minimum in the sense that any other state which differs from x at only a single point must have energy greater than or equal to $U(x|y)$.

Because ICM can not perform back-tracking, the initial condition is critical. A reasonable choice for an initial condition is the maximum likelihood estimate (MLE), or equivalently the estimate obtained by dropping the cost terms due to the *a priori* probability. Because the functions $l_s(y|\cdot)$ are only dependent on individual points in x, the MLE is easily written as

$$(\hat{x}_{MLE})_s = \arg\min_k \; l_s(y|k) \tag{10}$$

The algorithm for ICM may now be stated explicitly as follows:

1. Calculate $\hat{x}_{MLE}$ as an initial segmentation using (10).
2. Perform the local minimization defined by (9) at each pixel in a specified order.
3. If changes occur then repeat step 2, otherwise stop.

It should be noted that the specific order in which the replacements are performed is important. If raster scan ordering is used, current pixel replacements are affected by the classification of the adjacent pixels in the raster ordering. Since the prior distribution on X encourages pixels which are adjacent to be classified similarly, this often has the effect of biasing the position of boundaries in the direction of the raster ordering. It is not surprising that the solution obtained by ICM could be affected by the update ordering since the minimum reached is only local and may depend on details of the implementation. Another disadvantage of raster ordering is that it requires sequential replacements, and therefore does not allow for a parallel implementation.

The solution to these problems is to break x up into groupings such that the members of each grouping are conditionally independent of each other given the points in the remaining groups. Because of this property, any updating done within a group does not depend on ordering, and all points in a grouping may be updated in parallel. Such a division of x is known as a coding [5] and depends on the neighborhood system used. An eight-point neighborhood results in four groupings. So, it is possible to perform the updating in parallel using a maximum of $N/4$ processors.

An important aspect of a deterministic replacement algorithm such as ICM is that if the neighbors of a point have not changed then it need not be updated. This scheme can be implemented by keeping a flag at each point which indicates whether it is necessary to update the point. At the start of the algorithm all flags are set. Then each time a pixel is updated its flag is cleared and the neighboring flags are set if the pixel's value changed. Consequently, only the first iteration requires the application of the replacement operation at each point, and at later iterations most of the computation can be concentrated to regions where significant change is occurring. This can substantially reduce computation.

3.2 Multiple Resolution Segmentation (MRS)

In this paper, we shall perform segmentation using a multiple resolution approach. The MRS algorithm will begin segmentation at the coarsest resolution and use the result as an initial condition for segmentation at the next finer level of resolution. Each resolution will correspond to a level in a quad tree, so a lattice point at one resolution will correspond to four points at the next finer resolution. This group of four points will be referred to as a block, and $X^{(n)}$ will denote the field containing the classification of each of the blocks at resolution n. In general, quantities which vary as a function of resolution will have superscripts in parentheses to denote the level, and a superscript of 0 will denote the original quantity, so $X = X^{(0)}$. The mapping from pixel coordinates at the n^{th} resolution level to points at the previous coarser $(n+1)^{th}$ level can be formally written as

$$D(\,(i,j)\,) \;=\; (\,\lfloor i/2 \rfloor\,,\,\lfloor j/2 \rfloor\,)$$

where $\lfloor\cdot\rfloor$ denotes the greatest smaller integer function. Also, $D^k(s)$ will denote the composition of the function $D(\cdot)$ with itself k times. This structure is illustrated in Figure 1.

We may formulate local log likelihood functions for each resolution n by assuming that the fine segmentation, x, is an interpolated version of $x^{(n)}$. Specifically, if $\forall s \in S$, $x_s = x^{(n)}_{D^n(s)}$ then applying (4) results in

$$-\log f(y|x^{(n)}) = \sum_{s \in S^{(n)}} l^{(n)}_s(y|x^{(n)}_s)$$

where

$$\begin{aligned} l^{(n)}_s(y|k) &= \sum_{r \in D^{-1}(s)} l^{(n-1)}_r(y|k) \\ l^{(0)}_s(y|k) &= l_s(y|k)\ , \end{aligned} \tag{11}$$

and $D^{-1}(r)$ is the set of the points at the next finer resolution which map to r. The following section will discuss specific methods for computing these decimated likelihood functions.

At each resolution, our approach will be to solve an analogous segmentation problem to the one formulated for the fixed resolution problem. The optimization criteria we will use for finding the best segmentation at resolution n is

$$\hat{x}_{MAP}^{(n)} = \arg\min_x \left\{ \sum_{s\in S^{(n)}} l_s^{(n)}(y|x) + \beta_1^{(n)} t_1^{(n)}(x) + \beta_2^{(n)} t_2^{(n)}(x) \right\} , \tag{12}$$

where $t_1^{(n)}$ and $t_2^{(n)}$ count adjacent values of different type in $x^{(n)}$, and $\beta_1^{(n)}$ and $\beta_2^{(n)}$ are the parameters of the coarse resolution MRF. This criteria may be minimized by using the local replacement operation

$$\arg\min_k \left\{ l_s^{(n)}(y|k) + \beta_1^{(n)} v_1(x_{\partial s}^{(n)}, k) + \beta_2^{(n)} v_2(x_{\partial s}^{(n)}, k) \right\} . \tag{13}$$

The remaining question is how to choose these coarse resolution parameters. One possible approach is to choose these parameters so that the cost functions at coarse and fine resolutions are equal when $X_s = X_{D^n(s)}^{(n)}$ [11]. In fact, this may be done by observing that if $X_s^{(n-1)} = X_{D(s)}^{(n)}$ then

$$\begin{aligned} t_1^{(n-1)} &= 2\,t_1^{(n)} \\ t_2^{(n-1)} &= 2\,t_1^{(n)} + t_2^{(n)} . \end{aligned}$$

This relation implies that the fine and course resolution cost functions will be equal if

$$\begin{aligned} \beta_1^{(n)} &= 2(\beta_1^{(n-1)} + \beta_2^{(n-1)}) \\ \beta_2^{(n)} &= \beta_2^{(n-1)} \end{aligned} \tag{14}$$

for all resolutions n. By using these parameters, minimization of (12) corresponds to the minimization of the original cost criteria (7) under the constraint that the solution be constant on the appropriate sized blocks.

Coarse resolution minimization using the parameters given by (14) will effectively minimize (7) if the correct segmentation is approximately block constant. However, this recursion for the parameters has an undesirable property. It implies that the MRF models for coarser resolution segmentations should have progressively higher spatial correlation, or alternatively, finer resolution segmentations should have lower correlation. This, of course, runs counter to normal assumptions of spatial coherence in images, and will tend to cause insufficient spatial correlation at finer resolutions or excessive correlation at coarse resolutions.

A more reasonable approach is to assume that the spatial correlation is independent of the resolution since it avoids the problem of excessive correlation at coarse resolutions. Also, this assumption is appropriate when prior information is unavailable about the likely scale of regions in the image. Therefore, in all experimentation, we will fix the parameters of the MRF as a function of scale.

$$\begin{aligned} \beta_1^{(n)} &= \beta_1 \\ \beta_2^{(n)} &= \beta_2 \end{aligned}$$

The L level MRS algorithm may be defined as follows:

1. Compute $l_i^{(n)}(y|\cdot)$ for the levels $n = \{1, \ldots, L-1\}$ using (11).
2. Compute the ML estimate of $X^{(L-1)}$ as an initial condition using (10). Set $n = L-1$.
3. Compute $\hat{x}^{(n)}$ by iteratively performing the local minimization of (13) until no more changes occur.
4. If $n = 0$ stop. Otherwise, compute an initial value for $\hat{x}^{(n-1)}$ by replicating the values of $\hat{x}^{(n)}$ for each block in $\hat{x}^{(n-1)}$. Set $n = n-1$, and return to step 3.

3.3 Decimation of Likelihood Functions

A question now arises of how the decimated likelihood functions, $l_s^{(n)}(y|\cdot)$, should be calculated. The direct method is to explicitly calculate the functions for each texture, k, and sum those functions over the proper blocks. However, if the number of textures is large, this may require considerable computation and storage. Also, it presumes that the parameters of the textures are known. An alternative method is to extract statistics for each block so that the functions $l_s^{(n)}(y|k)$ may be computed both as a function of k and the model parameters θ_k. The parameters at different resolutions may then be calculated by decimation of those statistics from fine to coarse resolution.

Let $A \subset S$ be a region with N_A points over which the likelihood function is to be summed. Then the regions likelihood function, $l_A(y|k)$, may be expanded using (4) and (5).

$$l_A(y|k) = \sum_{s \in A} l_s(y|k)$$

$$
\begin{aligned}
&= \frac{1}{2}\sum_{s\in A}\left\{\frac{\tilde{y}_s^2}{\sigma_k^2} \;+\; \log(\sigma_k^2) \;+\; \log(2\pi)\right\} \\
&= \frac{1}{2}\left\{\frac{1}{\sigma_k^2}\sum_{s\in A}\tilde{y}_s^2 \;+\; N_A\log(\sigma_k^2) \;+\; N_A\log(2\pi)\right\}
\end{aligned}
$$

The parameters of the k^{th} texture model are $\theta_k = [\mu_k, \sigma_k^2, a_k]$. We may write the log likelihood as an explicit function of these parameters by defining three vectors. Let $\mathbf{a}_k$ be a vector of prediction coefficients in raster order. Let $\mathbf{y}_s$ be the vector of corresponding points in y, and let $\mathbf{1}$ be a vector of equal dimension with components of 1. For example, if $P = 3$ with a quarter plane prediction window the vectors have the following forms.

$$
\begin{aligned}
\mathbf{a}_k &= [1, \quad a_k(0,1), \quad a_k(1,0), \quad a_k(1,1)]^t \\
\mathbf{Y}_s &= [Y_s, \quad Y_{s-(0,1)}, \quad Y_{s-(1,0)}, \quad Y_{s-(1,1)}]^t \\
\mathbf{1} &= [1, \quad 1, \quad 1, \quad 1]^t
\end{aligned}
$$

The autocorrelation, R_A, and mean, m_A, statistics for the region A can be defined using these vectors.

$$
\begin{aligned}
m_A &= \frac{1}{N_A}\sum_{s\in A}\mathbf{Y}_s \qquad (15)\\
R_A &= \frac{1}{N_A}\sum_{s\in A}(\mathbf{Y}_s - m_A)(\mathbf{Y}_s - m_A)^t
\end{aligned}
$$

The prediction error may now be written as a function of these statistics.

$$
\begin{aligned}
\sum_{s\in A}\tilde{y}_s^2 &= \sum_{s\in A}\mathbf{a}_k^t(\mathbf{Y}_s - \mathbf{1}\mu_k)(\mathbf{Y}_s - \mathbf{1}\mu_k)^t\mathbf{a}_k \\
&= \sum_{s\in A}\left\{\mathbf{a}_k^t(\mathbf{Y}_s - m_k)(\mathbf{Y}_s - m_k)^t\mathbf{a}_k \;+\; \{\mathbf{a}_k^t(m_A - \mathbf{1}\mu_k)\}^2\right\} \\
&= N_A\left\{\mathbf{a}_k^t R_A \mathbf{a}_k \;+\; \{\mathbf{a}_k^t(m_A - \mathbf{1}\mu_k)\}^2\right\}
\end{aligned}
$$

By substituting this formula for the prediction error into the expression for $l_A(y|k)$, we may verify that R_A and m_A are sufficient statistics.

$$
l_A(y|k) = \frac{N_A}{2}\left\{\frac{\mathbf{a}_k^t R_A \mathbf{a}_k \;+\; \{\mathbf{a}_k^t(m_A - \mathbf{1}\mu_k)\}^2}{\sigma_k^2} \;+\; \log(\sigma_k^2) \;+\; \log(2\pi)\right\} \qquad (16)
$$

If we define a statistic vector $\xi_A = [N_A, R_A, m_A]$, we may define a new function, $L(\cdot,\cdot)$, such that

$$
L(\xi_A, \theta_k) \;=\; l_A(y|k) \;=\; \sum_{s\in A} l_s(y|k)\ . \qquad (17)
$$

Thus, if one can find a recursion for calculating the region statistics, $\xi_s^{(n)}$, from fine to coarse resolution, then the corresponding likelihood functions, $l_s^{(n)}(y|k)$, may be calculated from these statistics. Let A and B be two disjoint regions, and let $C = A \cup B$. Then, it may be shown using (15) that the statistics for C, ξ_C, may be calculated from ξ_A and ξ_B. The commutative operation of combining region statistics will be denoted by the symbol $\oplus$. Therefore

$$\xi_C = \xi_A \oplus \xi_B$$

where the components of ξ_C are

$$\begin{aligned}
N_C &= N_A + N_B \\
m_C &= \frac{N_A}{N_C} m_A + \frac{N_B}{N_C} m_B \\
R_C &= \frac{N_A}{N_C}\left\{R_A + (m_A - m_C)(m_A - m_C)^t\right\} + \frac{N_B}{N_C}\left\{R_B + (m_B - m_C)(m_B - m_C)^t\right\} .
\end{aligned}$$

The statistics at coarse resolutions may now be calculated from those at fine resolutions using this operation. If we use a notation for the repeated combining of region statistics, then an equation similar to (11) may be defined.

$$\begin{aligned}
\xi_r^{(n+1)} &= \bigoplus_{s \in D^{-1}(r)} \xi_s^{(n)} \\
&= \xi_{s_1}^{(n)} \oplus \xi_{s_2}^{(n)} \oplus \xi_{s_3}^{(n)} \oplus \xi_{s_4}^{(n)}
\end{aligned} \tag{18}$$

where $\{s_1, s_2, s_3, s_4\} = D^{-1}(r)$. This operation of combining region statistics will also be important in the agglomerative clustering used in unsupervised parameter estimation.

Finally, we will require a method for calculating the MLE estimate of the parameter vector θ for a region with statistic ξ. In order to simplify these calculations we will approximate the MLE estimate of μ by the average of the components of m. Therefore, $\hat{\mu} = \mathbf{n}^t m$ where $\mathbf{n}$ is a unit vector with equal components. If the autocorrelation is decomposed into the following components

$$\begin{bmatrix} r_{(0,0)} & b^t \\ b & Q \end{bmatrix} = R + (m - \mathbf{1}\hat{\mu})(m - \mathbf{1}\hat{\mu})^t ,$$

then the MLE values of the components of θ are given by

$$\begin{aligned}
\hat{\mu} &= \mathbf{n}^t m \\
\hat{a} &= [1, -b^t Q^{-1}]^t \\
\hat{\sigma}^2 &= a^t R a = r_{(0,0)} - b^t Q^{-1} b .
\end{aligned} \tag{19}$$

4 Unsupervised Parameter Estimation

In order to estimate the number and type of textures without supervision, an algorithm is needed which is independent of free parameters that may vary from image to image. The approach taken is to base the estimate on a statistical model of how much variation is likely among different texture samples. This results in an algorithm which is somewhat analogous to the well known K-means algorithm [17], but incorporates an additional step for determining the number of distinct textures.

4.1 Clustering Model

The first step in developing this clustering method is to calculate the joint probability distributions for the image and the pixel classifications given the texture parameters and the number of textures. The Gibbs distribution used in the previous section to model the pixel classifications can not be easily applied to the problem of unsupervised parameter estimation. The difficulty arises because the unsupervised estimation problem requires that the distribution of X be parameterized by the number of textures, M. However, the dependence on M is quite complex since the normalizing constant of the Gibbs distribution is a complex function of M. In fact, normalizing constants for Gibbs distributions have only been calculated in a few special cases (e.g. homogeneous Gaussian, Ising). This intractability introduces a significant problem since the normalizing constant is critical when comparing the likelihood of different clustering hypotheses. However, we would still like to use our prior knowledge about the spatial correlation of pixel classes in determining the number and types of textures present.

The approach we take is to first separate the image into blocks of equal size (normally at the coarsest resolution used in the MRS algorithm). The prior knowledge about the spatial correlation of pixel classes is then incorporated by assuming that each block contains only a single texture. Later, we will remove the effect of blocks which fall on boundaries and may contain a mixture of textures. In segmentation, we assumed that the classes of blocks had a Gibbs distribution in the form of (6) with properly chosen β parameters. The Gibbs distribution embodied the prior knowledge that adjacent blocks are likely to have the same texture. In this section, we ignore information regarding the spatial ordering of blocks. Conceptually, this ordering information may be ignored by assuming that the blocks have been randomly reordered. This is not an approximation since we may choose to ignore this information. However, our ability to distinguish similar textures will be diminished. The advantage of this approach is that the weakest possible assumptions are made about the prior distribution of the data, so that the clustering model may more accurately describe the observed data.

When the order of image blocks is ignored, the only relevant (sufficient) statistics are the texture statistics for each block, and the *number* of blocks of each texture. For fixed M, the Gibbs distribution determines the distribution of the number of blocks of each texture. However, for a general choice of M and β, a simple form for this distribution is not known. Therefore, we approximate this distribution using a multinomial distribution with unknown parameters. It should be emphasized that this is *not* equivalent to assuming that the ordered block classifications are independent random variables. In fact, block classifications may have high spatial correlation and still result in a multinomial distribution for the number of blocks of each texture.

All clustering will be done using texture statistics extracted from image blocks normally at the coarsest resolution level, $L-1$. Let $A_k \subset S^{(L-1)}$ be the set of blocks at this level which have a texture of type k for $k = 1, 2, \ldots, M$. Denote the number of elements in the set A_k as $W_k = |A_k|$, and the number of elements in $S^{(L-1)}$ as $W = |S^{(L-1)}|$. Notice that A_k will be random since it is a function of the random field $X^{(L-1)}$. W_k is in turn a random variable since it is a function of A_k. The sets, $\{A_k\}_{k=1}^{M}$, partition the field $X^{(L-1)}$ into groups each of which has uniform texture. Therefore, the random sets $\{A_k\}_{k=1}^{M}$ contain the same information as the random field $X^{(L-1)}$, and the two expressions can generally be interchanged in probability distributions. However, we have chosen to use the partitioning sets since they are suggestive of the clustering techniques to be developed. The additional constraints that $\bigcup_{k=1}^{M} A_k = S^{(L-1)}$ and $\sum_{k=1}^{M} W_k = W$, make the first $M-1$ values of A_k and W_k an unambiguous specification of the M^{th} value. Denote $\theta = [\theta_1, \ldots, \theta_M]^t$ as the vector of texture parameters.

In appendix A, the joint distribution of Y and the *unordered* block classes represented by $\{A_k\}_{k=1}^{M-1}$ is calculated. As described above, the the number of blocks of each texture, W_k, is modeled by a multinomial distribution with unknown parameters, ρ_k, and information about the relative position of blocks in the lattice is ignored. The distribution is then given by

$$p_{\theta\rho}(y, A_1, \ldots, A_{M-1}) = \prod_{k=1}^{M} \exp\left\{-L(\xi_{A_k}, \theta_k) + W_k \log \rho_k\right\} \tag{20}$$

where

$$\xi_{A_k} \;=\; \bigoplus_{s\in A_k} \xi_s \;,$$

the functions $L(\cdot,\cdot)$ are defined in (17) and the abbreviated notation ξ_s is used in place of $\xi_s^{(L-1)}$ to denote the statistic for block s. It is also shown in appendix A that by summing over all possible partitioning sets the distribution for y is given by

$$p_{\theta\rho}(y) = \prod_{s\in S}\left(\sum_{k=1}^{M} \rho_k \exp\left\{-L(\xi_s, \theta_k)\right\}\right) \tag{21}$$

which is recognizable as the distribution of W independent and identically distributed random variables with a marginal distribution formed by a mixture of M distributions.

While we now have an accurate closed form distribution as a function of the parameters θ and ρ, the problem of estimating the number of parameters remains poorly defined. For example, if the MLE texture parameters are estimated for the cases of both M and $M+1$ textures, then the fit of the $M+1$ texture case will always be as good or better since it includes the case of M textures. For this reason, ML estimation will generally choose a large or infinite number of textures to achieve the best fit to the observed data. However, this solution is unacceptable since segmentation is intended to partition the image into the minimum number of necessary distinct textures.

Zhang and Modestino [15, 16] have suggested that a reasonable alternative to the MLE is to find the number and type of regions which minimize the AIC information criteria proposed by Akaike[14]. The AIC contains an additional term to account for the bias due to the number of parameters to be estimated.

$$\text{AIC} = -2\log p_{\theta\rho}(y) \;+\; 2(\#\text{ identifiable parameters})$$

The condition of identifiability, which necessitates that the Fisher information matrix is invertible [18], is important in this application. Setting $\rho_M = 0$ would seem to result in a distribution with one less parameter. However, in this case the mean, variance and prediction coefficients corresponding to texture M are no longer identifiable. Therefore, the number of identifiable parameters is reduced by $P+3$ (where P is defined in section 2.1 as the number of AR prediction coefficients). Applying this definition of AIC to (21) yields

$$\text{AIC} = -2\log p_{\theta\rho}(y) \ + \ 2M(P+3) - 2 \ . \tag{22}$$

This optimization criteria has a number of disadvantages. Calculation of (22) requires the use of W statistics, one for each block. This contrasts with the calculation of (20) which only requires the use of M statistics, one for each texture. Minimization of (22) for fixed M is difficult due to the lack of known partitioning sets. Approximate minimization may be performed using the EM algorithm [19, 20] or equivalently Baum's algorithm [21]. However, the minimization process is still substantially complicated by the lack of class information, and does not result in an algorithm with a simple agglomerative clustering interpretation.

For these reasons, we propose a modified criteria which estimates the missing class information together with the number of textures and their parameters. This criteria results from replacing the log likelihood of y given in (21) with the joint log likelihood of y and the partitioning sets given in (20). This modified clustering criteria is

$$p_{\theta\rho}(y, A_1, \ldots, A_{M-1}) \ + \ 2M(P+3) - 2 \ . \tag{23}$$

The objective of all cluster techniques developed in the following sections will be to minimize this clustering criteria as a function of the parameters, $\{\theta, \rho\}$, the partitioning sets, $\{A_k\}_{k=1}^{M-1}$, and the number of textures, M.

Finally, we make a number of observations regarding properties of the clustering criteria given in (23). The manipulation of this criteria only requires the use of M statistics, and it will be seen that the resulting minimization techniques are analogous to traditional agglomerative clustering methods. By fixing the number of textures and the partitioning sets, minimization of this criteria yields the ML estimates of the parameters. By fixing the parameters and the number of textures, minimization yields the MAP estimate of the partitioning sets. The minimization of (23) is, of course, only unique up to a permutation of the texture parameters and the corresponding partitioning sets. We can make the solution unique by imposing an ordering on the parameter sets such as $\rho_k \leq \rho_{k+1}$. (If equality holds, then a finer ordering may be imposed using the θ_k parameters.) For a fixed M, the solution of this minimization problem may fall on a boundary of the parameter space corresponding to $\rho_M = 0$. This corresponds to the situation in which texture M never occurs and no blocks are classified with that texture. In this case, the optimum number of textures will be less than M since minimization for $M' \geq M$ will also result in $\rho_{M'} = 0$.

4.2 Estimation

The number of distinct textures and their parameters will be estimated using three procedures each of which strictly reduces the clustering criteria of (23). Each of the procedures may be thought of as minimizing with respect to one of M, (θ, ρ) or $\{A_k\}_{k=1}^M$ while fixing the remaining two quantities. The first procedure reestimates the parameters θ and ρ from the region statistics ξ_{A_k}. The second procedure repartitions the set S by choosing the sets $\{A_k\}_{k=1}^M$ which minimize (23). These two operations are analogous to the centroid estimation and data repartitioning in the K-means algorithm. In this context, each of the sets A_k contains the members of a cluster corresponding to texture k. The last procedure is an agglomerative hierarchical clustering algorithm which reduces M by combining pairs of sets A_k when this minimizes (23).

Because the terms involving θ and ρ can be separated in the log likelihood, the clustering criteria may be minimized by each variable independently. Minimizing with respect to θ is equivalent to calculating the ML estimate of θ_k with respect to ξ_{A_k} for each texture k. The algorithm for doing this is given in (19). The ML estimate of θ_k will be denoted by $\hat{\theta}(\xi_{A_k})$. The ML estimate of ρ may be found by minimizing (23) with respect to $\{\rho\}_{k=1}^M$ subject to the constraint that $\sum_{k=1}^M \rho_k = 1$. The result is

$$\hat{\rho}_k = \frac{W_k}{W} \ .$$

Choosing partitioning sets A_k is equivalent to choosing the field $x^{(L-1)}$ which minimizes (23) for a fixed set of parameters. This may be done by choosing the class of each block to be the most likely texture given the observed data.

$$\hat{x}_s^{(L-1)} = \arg\min_k \{L(\xi_s, \theta_k) - \log \rho_k\} \ .$$

Therefore, the best partition of the blocks is the partition which assigns each block to its most likely class.

$$\hat{A}_k = \left\{ s \in S^{(L-1)} : \forall l \ \ L(\xi_s, \theta_k) - \log \rho_k \leq L(\xi_s, \theta_l) - \log \rho_l \right\} \ .$$

A third mechanism is required to minimize (23) with respect to the number of clusters, M. The approach we take is to use agglomerative hierarchical clustering. Let $c(k,l)$ be the change in (23) caused by the merging of two clusters, A_k and A_l, into a single cluster, $C = A_k \cup A_l$. We again assume that the remaining parameters, (θ, ρ) and $\{A_j : j \neq k, l\}$, are held constant. However, in order to define $c(k,l)$ the parameters of the new cluster, C, must also be specified. We choose the ML estimate corresponding to C since this is the parameter vector which yields the minimum value of (23) after merging clusters. By using this definition of $c(k,l)$, we obtain the following expression

$$\begin{aligned} c(k,l) \;&=\; 2L(\xi_C, \hat{\theta}(\xi_C)) - 2L(\xi_{A_k}, \theta_k) - 2L(\xi_{A_l}, \theta_l) \\ &\quad + 2W_k \log\left(\frac{W_k}{W_k + W_l}\right) + 2W_l \log\left(\frac{W_l}{W_k + W_l}\right) - 2(P+3) \end{aligned} \tag{24}$$

where $\hat{\theta}(\xi_C)$ is the ML estimate of θ given the statistics $\xi_C = \xi_{A_k} \oplus \xi_{A_l}$ for the new cluster C. By substituting the MLE into (16), it is easily shown that for any statistic, ξ_A,

$$2L(\xi_A, \hat{\theta}(\xi_A)) = N_A \left\{1 + \log \hat{\sigma}_A^2 + \log(2\pi)\right\} \tag{25}$$

where $\hat{\sigma}_A^2$ is the ML estimate of the prediction error given the statistic ξ_A, and N_A is the number of pixels in A. In the complete clustering algorithm described at the end of this section, the functions $c(k,l)$ are only computed when the parameter values for clusters A_k and A_l take on their MLE values. Therefore, we may apply (25) to yield the following simplified expression for (24).

$$\begin{aligned} c(k,l) \;&=\; N_{A_k} \log\left(\frac{\hat{\sigma}_C^2}{\hat{\sigma}_{A_k}^2}\right) + N_{A_l} \log\left(\frac{\hat{\sigma}_C^2}{\hat{\sigma}_{A_l}^2}\right) \\ &\quad +2W_k \log\left(\frac{W_k}{W_k + W_l}\right) + 2W_l \log\left(\frac{W_l}{W_k + W_l}\right) - 2(P+3) \end{aligned} \tag{26}$$

From this formula, we can see that the only information required to calculate $c(k,l)$ is the statistic vector for the regions, ξ_{A_k} and ξ_{A_l}.

Since the global objective is to minimize (23), we are justified in merging any two clusters for which $c(k,l)$ is negative. This merging operation can be repeated until $c(k,l)$ is greater then or equal to zero for all clusters, A_k and A_l. In practice, it is often the case that many combinations of clusters have negative $c(k,l)$. Since the minimum found by this agglomeration procedure is not global, the order in which clusters are merged can affect the final result.

A steepest descent strategy first merges the two clusters with the minimum value of $c(k,l)$. A brute force approach to steepest descent requires M^2 calculations of $c(k,l)$ to find the two nearest clusters. However, the computational complexity of steepest descent may be reduced by using balanced trees [22]. For each cluster, A_k, a balanced tree is used to maintain a list of all other clusters in order of increasing $c(k,l)$. For each tree, insertions, deletions and searches may be performed in $\mathcal{O}(\log(M-1))$ time (since there are $M-1$ nodes in each tree). Therefore, the computation required to locate the two nearest clusters is equal to the number of trees multiplied by the computation required to search for the smallest element of each tree. This yields a total search time which is $\mathcal{O}(M\log(M-1))$. After two clusters have been merged a new tree must be created, and the remaining trees must be updated by deleting the old cluster entries and inserting the new entry. Together these operations require $\mathcal{O}(M\log M)$ computation. Therefore, the complete operation of combining two clusters requires $\mathcal{O}(M\log M)$ computation, and the total complexity of the agglomerative clustering procedure is $\mathcal{O}(M^2\log M)$. It should be noted that this algorithm requires $\mathcal{O}(M^2)$ memory.

Alternatively, a simpler algorithm which does not assure steepest descent may also be used. This algorithm, which is shown in Figure 2, starts by randomly ordering the clusters into a linked list. A cluster is then selected, and the list is searched to locate the nearest neighbor to the selected cluster. If $c(k,l)$ is negative, the clusters are combined to form a new cluster which replaces the original two clusters, and the new cluster is selected. Otherwise, a cluster is selected from a member of the list which has not been previously selected. This algorithm requires only $\mathcal{O}(M^2)$ computation, and $\mathcal{O}(M)$ memory.

A direct method for clustering the data is to start off with each image block as an individual cluster. However, the initial number of blocks (ranging from 1024 to 4096 in our experimental results) is often is too large to be easily handled. In addition, at this stage many cluster pairs result in a negative value for $c(k,l)$. Therefore, we first separate the data into quantization bins based on the mean and standard deviation of the sample statistics for a block. This efficiently groups together blocks which are very similar in their statistics. In all experiments we used 64 quantization levels for mean and 64 quantization levels for standard deviation for a total of 4096 distinct cluster types. This coarse grouping reduced the number of clusters to a range between 200 and 500.

After this initial grouping, all merging of clusters is done to strictly reduce the clustering criteria of (23). Since the initial number of clusters is large, the algorithm of Figure 2 is first used to merge all clusters with $c(k, l) < 10$. From this point on, only the steepest descent algorithm was used. This strategy is conservative since both these algorithms were found to yield similar and usually equivalent results when used individually.

An initial assumption of this analysis is that each block contains only a single texture. However, there will generally be blocks which fall on region boundaries and contain a mixture of textures. Zhang and Modestino have proposed a method of testing the variance of pixels in an image block and removing blocks that have excessively high variance [16]. However, in texture segmentation this method is unreliable since normal image textures are likely to have large variance. The approach we take is to perform ICM segmentation at the coarsest level of resolution using the estimated texture parameters. All blocks which fall on a texture boundary are then identified. A boundary block is defined as any block which differs in texture from one of its four nearest neighbors. All of these boundary blocks are then removed from their associated clusters. Cluster statistics and ML parameters are then recalculated and the clusters are merged using the steeped descent algorithm.

The complete algorithm for unsupervised parameter estimation is given below.

1 Extract texture statistics for each block of the image by using (18).

2 Group these statistics coarsely by mean and prediction variance in order to generate a tractable number of clusters.

3 Calculate ML parameter estimates for each cluster.

4 Combine clusters until no more negative values of $c(k, l)$ occur.

5 Calculate the cluster partitions, $\{A_k\}_{k=1}^{M}$, which minimize (23). If any changes have occurred return to step 3. Otherwise continue.

6 Perform ICM segmentation at the coarsest resolution, and remove from consideration any blocks which fall on boundaries. Recalculate the ML parameters, and then combine the remaining clusters until no more negative values of $c(k, l)$ occur.

5 Results

To verify the performance of the MRS algorithm, computer simulations were done under a variety of conditions which stressed the performance of the MRS, ICM and SA algorithms. The computational requirements of the segmentation algorithms were measured using two methods. The first method counts the number of visits per pixel at full resolution. In the case of MRS, this will usually be a fractional number since a visit at coarse resolution only represents a fraction of a visit at full resolution. However, this measure is very conservative for both deterministic algorithms since most local operations only require the testing of a flag (described at the end of section 3.1). The number of actual replacement computations per pixels is a more relevant measure of computation for implementations with significantly fewer computing elements than pixels.

All plots are in terms of the parameter λ which represents the cost per unit length of boundaries between differing regions.

$$\beta_1 = (\sqrt{2} - 1)\lambda = .414\,\lambda \quad \beta_2 = (\sqrt{2} - 1)\lambda/\sqrt{2} = .293\,\lambda$$

The constants relating λ and the parameters β are chosen to meet two constraints when used in conjunction with a square two dimensional lattice, S, with unit spacing between samples on both axes. First, the energy function $\beta_1 t_1 + \beta_2 t_2$ should be the same for diagonal, horizontal and vertical boundaries of equal length. Second, for these boundaries the energy function should equal the unit boundary length multiplied by λ. Both these constraints assume that boundaries are long (ignore end effects), and neither holds for boundaries which fall at angles other than diagonal, horizontal and vertical.

In all experiments, SA used an annealing schedule of the form

$$1/T(n) = 1/T(n-1) + \alpha \tag{27}$$

where $T(n)$ is the temperature used in the n^{th} pass. This annealing schedule is uniquely determined by the starting and ending temperatures and the number of iterations. In all experiments, the SA algorithm was started with the ML estimate of X as the initial condition.

There is some experimental evidence to believe that varying λ as a function of the number of iterations can improve performance in some applications of SA and ICM. For example, Besag [4] found that progressively increasing λ improved the performance of ICM in certain cases. Also, a number of studies have found that varying the parameters of the Gibbs distribution during ICM or SA can improve results when modeling continuously valued fields with discontinuities [10, 12]. However, in the case of continuous valued fields, the parameters were progressively reduced from initially large values. In any case, the schedule needed for changing these parameters introduces additional free parameters which generally are image dependent. Therefore, our approach is to fix the parameter λ when using ICM, or SA; and, as described earlier, to fix the parameter λ at all levels of resolution when using MRS.

A series of tests were performed on two simple 64x64 toroidal test patterns shown in Figure 3 to determine the relative performance of ICM, MRS and SA. In all these tests, three levels of resolution ($L = 2$) were used, and 100 interactions of SA were performed with $1/T(0) = 1$ and $1/T(100) = 3$. Figures 4 and 5 plot the mean number of misclassified pixels and the value of the energy (as defined in (7)) as a function of λ for MRS, ICM and SA. A set of plots is shown for two different cases each of which uses a test pattern shown in Figure 3 with added Gaussian noise. The signal-to-noise ratio was defined as

$$(S/N)_{dB} \;=\; 10 * \log((\mu_1 \;-\; \mu_2)^2/\sigma^2)\ .$$

The plots of mean error versus λ indicate that the performance of the MRS and SA algorithms are relatively insensitive to the choice of λ. The dynamic range of the energy function plotted in Figure 5 makes relative differences for pattern 1 difficult to distingish. The curves for SA and MR fall very close and effectively overlap in the plot.

Three additional tests were done for fixed λ to compare the performance of MRS, ICM and SA. The empirical density function of the error is shown in Figures 6, 7 and 8 together with the mean percentage error, the mean energy of the segmentation and the mean number of visits per pixel for each of the three methods. In all these examples, the MRS algorithm requires less computation than either ICM or SA. The empirical error distributions and the mean energy values indicate that the performance of MRS is comparable to SA and substantially better than ICM in some cases. In general, the advantages of MRS are greatest when the signal-to-noise ratio is low. In this case, the information needed to accurately classify pixels is spread over larger regions.

Segmentation was also performed for six different tests shown in Figures 9 to 14. In all but Figure 9, unsupervised parameter estimation was used. All test images have a resolution of 256x256, and all texture models used a three point ($P = 3$) quarter plane prediction model using the three closest pixels in the prediction. In each case, a free boundary was assumed with the parameter value $\lambda = 1.5$. When unsupervised parameter estimation was performed, it was done at the coarsest resolution used in segmentation. Each example contains segmentations resulting from MRS, ICM and SA for 100 and 500 iterations, a starting temperature of $T(0) = 1/0.8$ and an ending temperature of $T(1000) = 1/3.75$. When possible the correct segmentation was also presented. Each distinct texture is displayed as a unique gray level outlined in white.

Tables 1 and 2 summerize the results of the unsupervised estimation algorithm. Table 1 lists the number of distinct texture types estimated for each of the five experiments shown in Figures 10 to 14. The true number of textures is also listed for the three experiments in which the number of textures was known. In each case, the estimated number of textures was correct. Table 2 shows the actual and estimated parameter values for the synthetic image used in Figures 9 and 10. The actual parameters were those used in generating the synthetic textures.

Figures 9 to 12 used composite images formed by piecing together both natural and synthetic textures. In these composite images, adjacent textures were smoothly overlapped over a two pixel boundary in order to avoid a visible cue. In all four tests, four levels of resolution were used ($L = 3$). It should be noted that the unsupervised parameter algorithm requires at least one region of each texture large enough to guarantee that the region has an interior and therefore will be detected. The pattern for the texture regions has been chosen so this is true with four levels of resolution.

Figures 9 and 10 both used the same synthetic image. This composite image was formed from synthetic textures generated using the assumed texture model. Figure 9 shows the result of segmenting the synthetic image using the actual texture parameters, and Figure 10 shows the result of segmenting the same image using the parameters obtained from the unsupervised parameter estimation algorithm. Figures 11 and 12 used composite images formed from natural texture types. In order of their area, the first image contains woolen cloth, raffia and grass; and the second image contains woolen cloth, grass and beach sand.

Figures 13 and 14 each show the segmentation of an aerial photograph. Three levels of resolution (L=2) are used so that smaller regions with distinct textures could be identified. For these images, the quality of the segmentation depends on the application, and it is possible to vary the amount of detail in the segmentation by varying the boundary cost λ. Nevertheless, a superior algorithm should more reproducibly classify similar regions to the same texture.

Table 4 lists the value of the energy function resulting from each of the three minimization algorithms MRS, ICM and SA. The energy of the segmentation is useful since it provides an objective measure of performance. The computation required for segmentation using MRS, ICM and SA is shown in Table 3 both in terms of visits per pixel and replacements per pixel. In every case, the MRS algorithm required less computation than ICM and much less than SA. It should be noted that the number of replacements per pixel required by the MRS algorithm is relatively independent of the image segmented.

6 Conclusion

The MRS algorithm proposed in this paper performed better and required less computation than ICM in all cases tested. Also, the MRS algorithm generally performed comparably to SA but with substantially less computation. In fact, the computation required for SA ranged between 10 to 1000 times that of MRS depending on the problem and the method of comparison. However, the number of replacements per pixel was found to be a good basis for comparison, and using this measure SA typically required two orders of magnitude more computation for a range of examples. The MRS algorithm seems to yield the most substantial improvement in images when the information per pixel is low, so that information over larger regions must be combined to correctly segment the image.

An algorithm for performing unsupervised parameter estimation was proposed for use in conjunction with the segmentation algorithm. This algorithm is based on a Gaussian AR texture model and estimates both the number of textures with statistically significant differences and the parameters of those textures.

The algorithm is composed of three basic operations. The partitioning of data into distinct textures or clusters, the estimation of the cluster parameters and the agglomeration of similar clusters to reduce the number of distinct textures. All of these operations are performed in the context of minimizing a proposed statistical criteria, so the algorithm is guaranteed to converge. Superfluous textures formed at texture boundaries are removed in a final step.

A Appendix

In this section we will derive the joint distribution of Y and $\{A_k\}_{k=1}^{M-1}$ under the assumptions that the relative order of blocks in each set A_k is ignored, and that the number of blocks of each texture, W_k, have a multinomial distribution with unknown parameters, ρ_k. Using the function $L(\cdot,\cdot)$ defined in (17), we can rewrite the conditional distribution of Y given the partitioning sets $\{A_k\}_{k=1}^{M-1}$ explicitly as a function of the statistics, ξ_{A_k}, and the parameters θ_k. This yields the expression

$$\begin{aligned} f_\theta(y|A_1,\ldots,A_{M-1}) &= \exp\left\{-\sum_{s\in A_1} L(\xi_s,\theta_1) - \ldots - \sum_{s\in A_M} L(\xi_s,\theta_M)\right\} \\ &= \prod_{k=1}^{M} \exp\left\{-L(\xi_{A_k},\theta_k)\right\} \end{aligned} \tag{28}$$

where

$$\xi_{A_k} = \bigoplus_{s\in A_k} \xi_s \ .$$

One method of conceptualizing the loss of ordering information is to assume that the values in $X^{(L-1)}$ and the corresponding statistics, ξ_s, have been reordered with a random mapping that is uniformly distributed over all possible reorderings. In this case, any unique mapping occurs with equal probability $1/W!$. If Π_s is the random reordering, then

$$A_k = \{s : X_{\Pi_s}^{(L-1)} = k\} \ .$$

If we condition on the knowledge of $\{W_k\}_{k=1}^{M-1}$ and the value of $X^{(L-1)}$, then the probability of any particular group of partitioning sets is given by the sum of the probabilities of all mappings which would result in that partitioning. Since there are exactly $\prod_{k=1}^{M} W_k!$ unique mappings which result in any group of partitioning sets, we have that

$$\begin{aligned} p(A_1,\ldots,A_{M-1}|X,\ W_1,\ldots,W_{M-1}) &= \frac{\prod_{k=1}^{M} W_k!}{W!} \\ &= p(A_1,\ldots,A_{M-1}|W_1,\ldots,W_{M-1}) \ . \end{aligned}$$

We next assume that $\{W_k\}_{k=1}^{M-1}$ have a multinomial distribution with unknown parameters $\rho = [\rho_1, \ldots, \rho_{M-1}]^t$ and $\rho_M = 1 - \sum_{k=1}^{M-1} \rho_k$, then

$$\begin{aligned} p_\rho(A_1, \ldots, A_{M-1}) &= p(A_1, \ldots, A_{M-1} | W_1, \ldots, W_{M-1}) p_\rho(W_1, \ldots, W_{M-1}) \\ &= \prod_{k=1}^{M} \rho_k^{W_k} \end{aligned} \tag{29}$$

By combining (28) with (29) the final expression of the joint distribution of Y and $\{A_k\}_{k=1}^{M-1}$ results.

$$p_{\theta\rho}(y, A_1, \ldots, A_{M-1}) = \prod_{k=1}^{M} \exp\left\{-L(\xi_{A_k}, \theta_k) + W_k \log \rho_k\right\} \tag{30}$$

The distribution for Y can now be computed by summing over all possible partitioning sets, or equivalently, by summing over all possible values of x. To do this, we let the sequence $\{s_1, s_2, \ldots, s_N\}$ be some ordering of all the points in S. Then we have

$$\begin{aligned} p_{\theta\rho}(y) &= \sum_{x_{s_1}=1}^{M} \cdots \sum_{x_{s_N}=1}^{M} p_{\theta\rho}(y, x_{s_1}, \ldots, x_{s_N}) \\ &= \sum_{x_{s_1}=1}^{M} \cdots \sum_{x_{s_N}=1}^{M} \prod_{i=1}^{N} \rho_{x_{s_i}} \exp\{-l_{s_i}(y|x_{s_i})\} \\ &= \sum_{x_{s_1}=1}^{M} \cdots \sum_{x_{s_{N-1}}=1}^{M} \prod_{i=1}^{N-1} \rho_{x_{s_i}} \exp\{-l_{s_i}(y|x_{s_i})\} \left[\sum_{x_{s_N}=1}^{M} \rho_{x_{s_N}} \exp\{-l_{s_N}(y|x_{s_N})\} \right] \\ &= \sum_{x_{s_1}=1}^{M} \cdots \sum_{x_{s_{N-1}}=1}^{M} \prod_{i=1}^{N-1} \rho_{x_{s_i}} \exp\{-l_{s_i}(y|x_{s_i})\} \left[\sum_{k=1}^{M} \rho_k \exp\{-L(\xi_{s_N}, \theta_k)\} \right] \end{aligned}$$

where x_s is used in place of $x_s^{(L-1)}$. By exchanging products and summations, we then obtain

$$p_{\theta\rho}(y) = \prod_{s \in S} \left(\sum_{k=1}^{M} \rho_k \exp\left\{-L(\xi_s, \theta_k)\right\} \right) \tag{31}$$

which is now recognizable as the distribution of W independent and identically distributed random variables with a marginal distribution formed by a mixture of M distributions.

References

[1] H. Derin and H. Elliott, "Modeling and Segmentation of Noisy and Textured Images Using Gibbs Random Fields," *IEEE Trans. Pat. An. Mach. Intell.,* vol. PAMI-9, pp. 39-55, Jan. 1987.

[2] C. S. Won and H. Derin, "Segmentation of Noisy Textured Images Using Simulated Annealing," *Proceedings: ICASSP 87,* pp. 14.4.1-14.4.4, Dallas, TX 1987.

[3] H. Derin and C. S. Won, "A Parallel Image Segmentation Algorithm Using Relaxation with Varying Neighborhoods and Its Mapping to Array Processors," *Comput. Vision Graphics and Image Process.,* vol. 40, pp. 54-78, Oct. 1987.

[4] J. Besag, "On the Statistical Analysis of Dirty Pictures," *J. Roy. Statist. Soc. B,* vol. 48, No. 3, pp. 259-302, 1986.

[5] J. Besag, "Spatial Interaction and the Statistical Analysis of Lattice Systems", *J. Roy. Statist. Soc. B,* vol. 36, No. 2, pp. 192-236, 1974.

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

[7] A. Jain, "Advances in Mathematical Models for Image Processing" *Proc. of the IEEE,* vol. 69, pp. 502-528, May 1981.

[8] R. Kashyap and R. Chellappa, "Estimation and Choice of Neighbors in Spatial-Interaction Models of Images," *IEEE Trans. Inform. Theory,* vol. IT-29, pp. 60-72, Jan. 1983.

[9] N. Metropolis, *et. al.,* "Equations of State Calculations by Fast Computing Machines," *J. Chem. Phys.,* vol. 21, pp. 1087-1091, 1953.

[10] J. Hutchinson, C. Koch, J. Luo and C. Mead, "Computing Motion Using Analog and Binary Resistive Networks," *Computer,* vol. 21, pp. 53-63, March 1988.

[11] C. Bouman and Bede Liu, "Segmentation of Textured Images Using a Multiple Resolution Approach," *Proc. IEEE Int'l Conf. on Acoust., Speech and Sig. Proc.*, pp. 1124-1127, New York, NY, April 11-14, 1988.

[12] C. Bouman and Bede Liu, "A Multiple Resolution Approach to Regularization," *Proc. SPIE Conf. on Visual Comm. and Image Proc.*, pp. 512-520, Cambridge, MA, Nov. 9-11, 1988.

[13] D. Terzopoulos, "Image Analysis Using Multigrid Relaxation Methods," *IEEE Trans. Pat. An. Mach. Intell.,* vol. PAMI-8, pp. 129-139, March 1986.

[14] H. Akaike, "A New Look at the Statistical Model Identification," *IEEE Trans. Automat. Contr.,* vol. AC-19, pp. 716-723, Dec. 1974.

[15] J. Zhang and J. W. Modestino, "A Model-Fitting Approach to Cluster Validation with Application to Stochastic Model-Based Image Segmentation," *Proceedings: ICASSP 88*, New York, NY 1988.

[16] J. Zhang and J. W. Modestino, "Unsupervised Image Segmentation Using A Gaussian Model," *Proceedings of the 1988 Conference on Information Sciences and Systems,* Princeton, NJ 1988.

[17] R. Duda and P. Hart, *Pattern Classification and Scene Analysis,* Wiley, New York, NY, 1973.

[18] S. D. Silvey, *Statistical Inference,* Chapman and Hall, London, 1975.

[19] A. Dempster, N. Laird and D. Rubin, "Maximum Likelihood from Incomplete Data via the EM Algorithm," *J. Roy. Statist. Soc. B,* vol. 39, no. 1, pp. 1-38, 1977.

[20] E. Redner and H. Walker, "Mixture Densities, Maximum Likelihood and the EM Algorithm," *SIAM Review,* vol. 26, no. 2, April 1984.

[21] L. Baum, T. Petrie, G. Soules, N. Weiss, "A Maximization Technique Occurring in the Statistical Analysis of Probabilistic Functions of Markov Chains," *Ann. Math. Statistics,* vol. 41, no. 1, pp. 164-171, 1970.

[22] D. E. Knuth, *The Art of Computer Programming: Volume 3 / Sorting and Searching,* Addision-Wesley, Reading, Mass., 1973.

image type	number of textures	
	actual	estimated
synthetic AR	3	3
natural textures #1	3	3
natural textures #2	3	3
aerial photo #1	N/A	4
aerial photo #2	N/A	6

Table 1: Number of distinct textures estimated for each of the five experiments which used unsupervised parameter estimation.

		mean	prediction variance	prediction coefficients
actual	texture 1	127.0	64.0	[-0.850, -0.850, 0.723]
	texture 2	127.0	81.0	[-0.700, -0.700, 0.420]
	texture 3	127.0	16.0	[-0.735, -0.757, 0.557]
estimated	texture 1	125.6	63.5	[-0.851, -0.847, 0.720]
	texture 2	130.3	82.2	[-0.710, -0.695, 0.423]
	texture 3	128.9	15.7	[-0.688, -0.785, 0.546]

Table 2: True and estimated parameters for the three textures of the synthetic image.

image type	visits per pixel			replacements per pixel		
	MRS	ICM	SA	MRS	ICM	SA
synthetic AR	7.61	27.00	100, 500	1.39	2.96	100, 500
unsupervised AR	7.80	29.00	$\downarrow$	1.39	3.00	$\downarrow$
natural textures #1	7.55	21.00		1.42	2.11	
natural textures #2	9.61	13.00		1.44	1.90	
aerial photo #1	12.69	13.00		1.48	2.34	
aerial photo #2	10.75	13.00		1.64	2.12	

Table 3: Number of visits per pixel and number of replacements per pixel for MRS, ICM and SA. Experiments using SA were perform with both 100 and 500 iteration annealing schedules.

image type	energy of segmentation			
	MRS	ICM	SA 100	SA 500
synthetic AR	165319	180499	165438	164970
unsupervised AR	165310	179630	165376	164988
natural textures #1	198891	203057	198850	198658
natural textures #2	213437	214801	212997	212968
aerial photo #1	164851	168909	164256	164184
aerial photo #2	110826	114738	109948	109723

Table 4: Total energy of the segmentation for each of the six experiments.

MRS

Figure 1: The structure of the multiresolution algorithm is given by a quadtree where a point at coarse resolution corresponds to four point at the next finer resolution.

.

```
$k \leftarrow$ List
While($k \neq$NULL) {
    Min $= \infty$
    $l \leftarrow$ List
    While($l \neq$NULL) {
        if( $(c(k,l) <$ Min$)\&(l \neq k)$ ) {
            Min $\leftarrow c(k,l)$
            Save $\leftarrow l$
        }
        $l \leftarrow$ NEXT($l$)
    }
    if(Min$< 0$) {
        CLUSTER($k$)$\leftarrow$ CLUSTER($k$)$\oplus$CLUSTER(Save)
        DELETE(Save)
    }
    else $k \leftarrow$ NEXT($k$)
}
```

Figure 2: Pseudocode for a clustering strategy which searches for a minimum of the clustering criteria. *List* is a pointer to a linked list containing information about each cluster.

Test Patterns

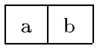

Figure 3: Binary 64x64 resolution test patterns used to evaluate the performance of MRS, ICM and SA segmentation algorithms. a) pattern 1 b) pattern 2.

.

Error Plot

Figure 4: The mean percentage error is plotted versus the boundary cost parameter λ for two cases. Each point was computed from the average of 100 samples.

Energy Plot

Figure 5: The mean energy of the segmentation is plotted versus the boundary cost parameter λ for two cases. Each point was computed from the average of 100 samples.

Density1

Figure 6: Empirical density of the percentage error resulting from the segmentation of test pattern 2 at 0 dB signal-to-noise ratio. Each density function was computed from 1000 samples.

Density2

Figure 7: Empirical density of the percentage error resulting from the segmentation of test pattern 1 at 0 dB signal-to-noise ratio. Each density function was computed from 1000 samples.

Density3

Figure 8: Empirical density of the percentage error resulting from the segmentation of test pattern 1 at -6 dB signal-to-noise ratio. Each density function was computed from 1000 samples.

Supervised synthetic segmentation

a	b	c
d	e	f

Figure 9: *Techniques using true parameter values:* a) Synthetic image obtained by superimposing three nonoverlapping Gaussian AR textures. b) Correct segmentation of image. Segmentations resulting from c) MRS algorithm, d) ICM algorithm, e) 100 iterations of SA and f) 500 iterations of SA.

Unsupervised synthetic segmentation

a	b	c
d	e	f

Figure 10: *Techniques using unsupervised parameter estimation:* a) Synthetic image obtained by superimposing three nonoverlapping Gaussian AR textures. Three textures result from unsupervised estimation. b) Correct segmentation of image. Segmentations resulting from c) MRS algorithm, d) ICM algorithm, e) 100 iterations of SA and f) 500 iterations of SA.

Quilt1

a	b	c
d	e	f

Figure 11: a) Composit of natural textures #1. The image is composed of wool, raffia and grass. Three textures result from unsupervised estimation. b) Correct segmentation of image. Segmentations resulting from c) MRS algorithm, d) ICM algorithm, e) 100 iterations of SA and f) 500 iterations of SA.

Quilt2

a	b	c
d	e	f

Figure 12: a) Composit of natural textures #2. The image is composed of wool, grass and beach sand. Three textures result from unsupervised estimation. b) Correct segmentation of image. Segmentations resulting from c) MRS algorithm, d) ICM algorithm, e) 100 iterations of SA and f) 500 iterations of SA.

Aerial 16d2

Figure 13: a) Aerial photograph #1. Four textures result from unsupervised estimation. Segmentations resulting from b) MRS algorithm, c) ICM algorithm, d) 100 iterations of SA and e) 500 iterations of SA.

Aerial 3d1

Figure 14: a) Aerial photograph #2. Six textures result from unsupervised estimation. Segmentations resulting from b) MRS algorithm, c) ICM algorithm, d) 100 iterations of SA and e) 500 iterations of SA.