CHAPTER 10

IMAGE SEGMENTATION

Segmentation is an important step in medical image analysis and classification for radiological evaluation or computer-aided diagnosis. Image segmentation refers to the process of partitioning an image into distinct regions by grouping together neighborhood pixels based on some predefined similarity criterion. The similarity criterion can be determined using specific properties or features of pixels representing objects in the image. In other words, segmentation is a pixel classification technique that allows the formation of regions of similarities in the image.

Image segmentation methods can be broadly classified into three categories:

1. Edge-based methods in which the edge information is used to determine boundaries of objects. The boundaries are then analyzed and modified as needed to form closed regions belonging to the objects in the image.
2. Pixel-based methods in which heuristics or estimation methods derived from the histogram statistics of the image are used to form closed regions belonging to the objects in the image.
3. Region-based methods in which pixels are analyzed directly for a region-expansion process based on a predefined similarity criterion to form closed regions belonging to the objects in the image.

Once the regions are defined, features can be computed to represent regions for characterization, analysis, and classification. These features may include shape and texture information of the regions as well as statistical properties, such as variance and mean of gray values.

This chapter describes major image segmentation methods for medical image analysis and classification.

10.1. EDGE-BASED IMAGE SEGMENTATION

Edge-based approaches use a spatial filtering method to compute the first-order or second-order gradient information of the image. As described in Chapter 9, Sobel or other directional derivative masks can be used to compute gradient information. The Laplacian mask can be used to compute second-order gradient information of the image. For segmentation purposes, edges need to be linked to form closed regions. Gradient information of the image is used to track and link relevant edges.

Medical Image Analysis, *Second Edition*, by Atam P. Dhawan

The second step is usually very tedious. In addition, it has to deal with the uncertainties in the gradient information due to noise and artifacts in the image.

10.1.1 Edge Detection Operations

The gradient magnitude and directional information from the Sobel horizontal and vertical direction masks can be obtained by convolving the respective G_x and G_y masks with the image as (1, 2)

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

$$G_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

$$M = \sqrt{G_x^2 + G_y^2} \approx |G_x| + |G_y| \tag{10.1}$$

where M represents the magnitude of the gradient that can be approximated as the sum of the absolute values of the horizontal and vertical gradient images obtained by convolving the image with the horizontal and vertical masks G_x and G_y.

The second-order gradient operator Laplacian can be computed by convolving one of the following masks, $G_{L(4)}$ and $G_{L(8)}$, which, respectively, use a 4- and 8-connected neighborhood.

$$G_{L(4)} = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

$$G_{L(8)} = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}. \tag{10.2}$$

The second-order derivative, Laplacian, is very sensitive to noise as can be seen from the distribution of weights in the masks in Equation 10.2. The Laplacian mask provides a nonzero output even for a single pixel-based speckle noise in the image. Therefore, it is usually beneficial to apply a smoothing filter first before taking a Laplacian of the image. The image can be smoothed using a Gaussian weighted spatial averaging as the first step. The second step then uses a Laplacian mask to determine edge information. Marr and Hildreth (3) combined these two steps into a single Laplacian of Gaussian (LOG) function as

$$\begin{aligned} h(x, y) &= \nabla^2[g(x, y) \otimes f(x, y)] \\ &= \nabla^2[g(x, y)] \otimes f(x, y) \end{aligned} \tag{10.3}$$

where $\nabla^2[g(x, y)]$ is the LOG function that is used for spatial averaging and is commonly expressed as the Mexican hat operator:

$$\nabla^2[g(x, y)] = \left(\frac{x^2 + y^2 - 2\sigma^2}{\sigma^4} \right) e^{-\frac{(x^2+y^2)}{2\sigma^2}} \tag{10.4}$$

where σ^2 is the variance of the Gaussian function.

A LOG mask for computing the second-order gradient information of the smoothed image can be computed from Equation 10.4. With $\sigma = 2$, the LOG mask G_{LOG} of 5×5 pixels is given by

$$G_{LOG} = \begin{bmatrix} 0 & 0 & -1 & 0 & 0 \\ 0 & -1 & -2 & -1 & 0 \\ -1 & -2 & 16 & -2 & -1 \\ 0 & -1 & -2 & -1 & 0 \\ 0 & 0 & -1 & 0 & 0 \end{bmatrix}. \tag{10.5}$$

The image obtained by convolving the LOG mask with the original image is analyzed for zero crossing to detect edges since the output image provides values from negative to positive. One simple method to detect zero crossing is to threshold the output image for zero value. This operation provides a new binary image such that a "0" gray value is assigned to the binary image if the output image has a negative or zero value for the corresponding pixel. Otherwise, a high gray value (such as "255" for an 8 bit image) is assigned to the binary image. The zero crossing of the output image can now be easily determined by tracking the pixels with a transition from black ("0" gray value) to white ("255" gray value) on an 8-bit gray-level scale.

10.1.2 Boundary Tracking

Edge detection operations are usually followed up by the edge-linking procedures to assemble meaningful edges to form closed regions. Edge-linking procedures are based on pixel-by-pixel search to find connectivity among the edge segments. The connectivity can be defined using a similarity criterion among edge pixels. In addition, geometrical proximity or topographical properties are used to improve edge-linking operations for pixels that are affected by noise, artifacts, or geometrical occlusion. Estimation methods based on probabilistic approaches, graphs, and rule-based methods for model-based segmentation have also been used (4–27).

In neighborhood search methods, the simplest method is to follow the edge detection operation by a boundary-tracking algorithm. Let us assume that the edge detection operation produces edge magnitude $e(x, y)$ and edge orientation $\phi(x, y)$ information. The edge orientation information can be directly obtained from the directional masks, as described in Chapter 9, or computed from the horizontal and vertical gradient masks. Let us start with a list of edge pixels that can be selected by scanning the gradient image obtained from the edge detection operation. Assuming the first edge pixel as a boundary pixel b_j, a successor boundary pixel b_{j+1} can be found in the 4- or 8-connected neighborhood if the following conditions are satisfied:

$$
\begin{aligned}
&|e(b_j)| > T_1 \\
&|e(b_{j+1})| > T_1 \\
&|e(b_j) - e(b_{j+1})| < T_2 \\
&|\phi(b_j) - \phi(b_{j+1})| \bmod 2\pi < T_3
\end{aligned} \tag{10.6}
$$

where T_1, T_2, and T_3 are predetermined thresholds.

If there is more than one neighboring pixel that satisfies these conditions, the pixel that minimizes the differences is selected as the next boundary pixel. The algorithm is recursively applied until all neighbors are searched. If no neighbor is found satisfying these conditions, the boundary search for the striating edge pixel is stopped and a new edge pixel is selected. It can be noted that such a boundary-tracking algorithm may leave many edge pixels and partial boundaries unconnected. Some a priori knowledge about the object's boundaries is often needed to form regions with closed boundaries. Relational tree structures or graphs can also be used to help form the closed regions (28, 29).

A graph-based search method attempts to find paths between the start and end nodes minimizing a cost function that may be established based on the distance and transition probabilities. The start and end nodes are determined by scanning the edge pixels based on some heuristic criterion. For example, an initial search may label the first edge pixel in the image as the start node and all the other edge pixels in the image or a part of the image as potential end nodes. Among several graph-based search algorithms, the A* algorithm is widely used (28, 29).

The A* search algorithm can be implemented using the following steps (29):

1. Select an edge pixel as the start node of the boundary and put all of the successor boundary pixels in a list, OPEN.
2. If there is no node in the OPEN list, stop; otherwise continue.
3. For all nodes in the OPEN list, compute the cost function $t(z_k)$ and select the node z_k with the smallest cost. Remove the node z_k from the OPEN list and label it as CLOSED. The cost function $t(z)$ may be computed as

$$
\begin{aligned}
t(z_k) &= c(z_k) + h(z_k) \\
c(z_k) &= \sum_{i=2}^{k} s(z_{i-1}, z_i) + \sum_{j=1}^{k} d(z_j)
\end{aligned} \tag{10.7}
$$

   where $s(z_{i-1}, z_i)$ and $d(z_i)$ are the transition and local costs, and $h(z_k)$ is the lower bound estimate of the cost function.
4. If z_k is the end node, exit with the solution path by backtracking the pointers; otherwise continue.
5. Expand the node z_k by finding all successors of z_i. If there is no successor, go to step 2; otherwise continue.
6. If a successor z_i is not labeled yet in any list, put it in the list OPEN with updated cost as $c(z_i) = c(z_k) + s(z_k, z_i) + d(z_i)$ and a pointer to its predecessor z_k.

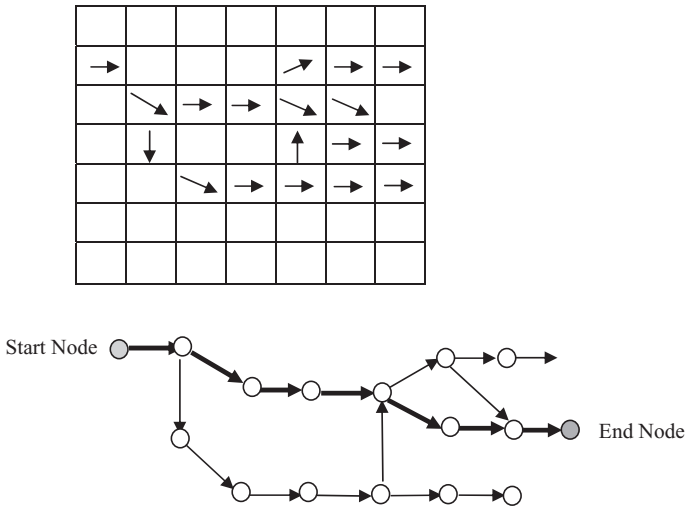


Figure 10.1 Top: An edge map with magnitude and direction information. Bottom: A graph derived from the edge map with a minimum cost path (dark arrows) between the start and end nodes.

7. If a successor z_i is already labeled as CLOSED or OPEN, update its value by $c'(z_i) = \min[c(z_i), c(z_k) + s(z_k, z_i)]$. Put those CLOSED successors, whose cost functions $c'(z_i)$ were lowered, in the OPEN list and redirect to z_k the pointers from all nodes whose costs were lowered.
8. Go to step 2.

The A* algorithm may not always find the optimal solution with minimum cost path a suboptimal solution can be found with the use of some a priori or heuristic information (29). Figure 10.1 shows a schematic representation of an example edge map and a graph of edge nodes with a probable minimum cost path. It can be seen that the path marked with bold arrows may not necessarily be the optimal path.

10.1.3 Hough Transform

The Hough transform is used to detect straight lines and other parametric curves such as circles and ellipses (14). It can also be used to detect boundaries of an arbitrarily shaped object if the parameters of the object are known. The basic concept of the generalized Hough transform is that an analytical function such as a straight line, a circle, or a closed shape, represented in the image space (spatial domain), has a dual representation in the parameter space. For example, the general equation of a straight line can be given as

$$y = mx + c \tag{10.8}$$

where m is the slope and c is the y-intercept.

As can be seen from Equation 10.8, the locus of points is described by two parameters, slope and y-intercept. Therefore, a line in the image space forms a point

(m, c) in the parameter space. Likewise, a point in the image space forms a line in the parameter space. Therefore, a locus of points forming a line in the image space will form a set of lines in the parameter space, whose intersection represents the parameters of the line in the image space. If a gradient image is thresholded to provide edge pixels, each edge pixel can be mapped to the parameter space. The mapping can be implemented using the bins of points in the parameter space. For each edge pixel of the straight line in the image space, the corresponding bin in the parameter space is updated. At the end, the bin with the maximum count represents the parameters of the straight line detected in the image. The concept can be extended to map and detect boundaries of a predefined curve. In general, the points in the image space become hyperplanes in the N-dimensional parameter space and the parameters of the object function in the image space can be found by searching the peaks in the parameter space caused by the intersection of the hyperplanes.

To detect object boundaries using the Hough transform, it is necessary to create a parameter model of the object. The object model is transferred into a table called an R-table. The R-table can be considered as a one-dimensional array where each entry of the array is a list of vectors. For each point in the model description (MD), a gradient along with the corresponding vector extending from the boundary point to the centroid (Fig. 10.2) is computed. The gradient acts as an index for the R-table.

For object recognition, a two-dimensional (2-D) parameter space of possible x–y coordinate centers is initialized, with the accumulator values associated with each location set to zero. An edge pixel from the gradient image is selected. The gradient information is indexed into the R-table. Each vector in the corresponding list is added to the location of the edge pixel. The endpoint of the vector should now point to a new edge pixel in the gradient image. The accumulator of the corresponding location in the parameter space is then increased by one. As each edge pixel is examined, the accumulator for the corresponding location receives the highest count. If the model object is considered to be translated in the image, the accumulator for the correct translation location would receive the highest count. To deal with rotation and scaling, the process must be repeated for all possible rotations and scales. Thus, the complete process could become very tedious if a large number of rotations and

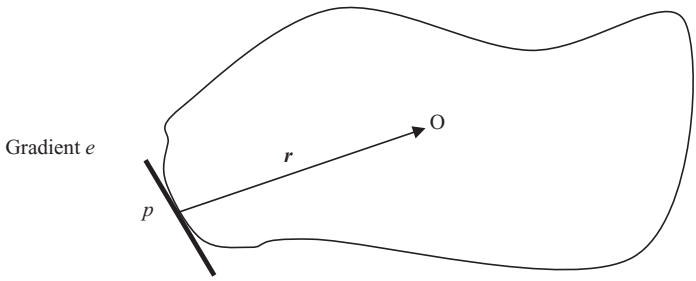


Figure 10.2 A model of an object shape to be detected in the image using Hough transform. The vector $\boldsymbol{r}$ connects the centroid and a tangent point p. The magnitude and angle of the vector $\boldsymbol{r}$ are stored in the R-table at a location indexed by the gradient of the tangent point p.

scales are examined. To avoid this complication, simple transformations can be made in the *R*-table of the transformation (14).

10.2. PIXEL-BASED DIRECT CLASSIFICATION METHODS

The pixel-based direct classification methods use histogram statistics to define single or multiple thresholds to classify an image pixel by pixel. The threshold for classifying pixels is obtained from the analysis of the histogram of the image. A simple approach is to examine the histogram for bimodal distribution. If the histogram is bimodal, the threshold can be set to the gray value corresponding to the deepest point in the histogram valley. If not, the image can be partitioned into two or more regions using some heuristics about the properties of the image. The histogram of each partition can then be used for determining thresholds. By comparing the gray value of each pixel to the selected threshold, a pixel can be classified into one of two classes.

Let us assume that an image or a part of an image has a bimodal histogram of gray values as shown in Figure 10.3. The image $f(x, y)$ can be segmented into two classes using a gray value threshold T such that

$$g(x, y) = \begin{cases} 1 & \text{if } f(x, y) > T \\ 0 & \text{if } f(x, y) \leq T \end{cases} \tag{10.9}$$

where $g(x, y)$ is the segmented image with two classes of binary gray values, "1" and "0", and T is the threshold selected at the valley point from the histogram.

A simple approach to determine the gray-value threshold T is by analyzing the histogram for the peak values and then finding the deepest valley point between the two consecutive major peaks. If a histogram is clearly bimodal, this method gives good results. However, medical images may have multiple peaks with specific requirements of regions to be segmented. For example, the magnetic resonance (MR) brain image shown in Figure 10.3 has two major peaks in the histogram. The first major peak is at the gray value 0, representing the black background. The second major peak is at the gray value 71, with a distribution of gray values corresponding to the brain tissue regions within the skull. The first requirement of the MR brain image segmentation may be to extract the overall brain area under the skull. This can be accomplished by finding out the deepest valley point between the two major peaks. The threshold T of gray value 12 is thus selected to separate the brain area from the background as shown in Figure 10.3. It can be noted that there are some holes within the segmented brain region using the gray-value threshold. These holes may be filled to quantify the overall brain area under the skull in the image.

To segment specific brain regions such as ventricles (the large structure in the middle of the image), additional thresholds can be determined from the histogram. These thresholds can be determined by finding out additional peaks in the remaining distribution of the histogram belonging to the second major peak, that is, around or after the gray value 12. A close examination of the histogram in Figure 10.3 shows a third peak around the gray value 254, representing the brightest pixels in the image.

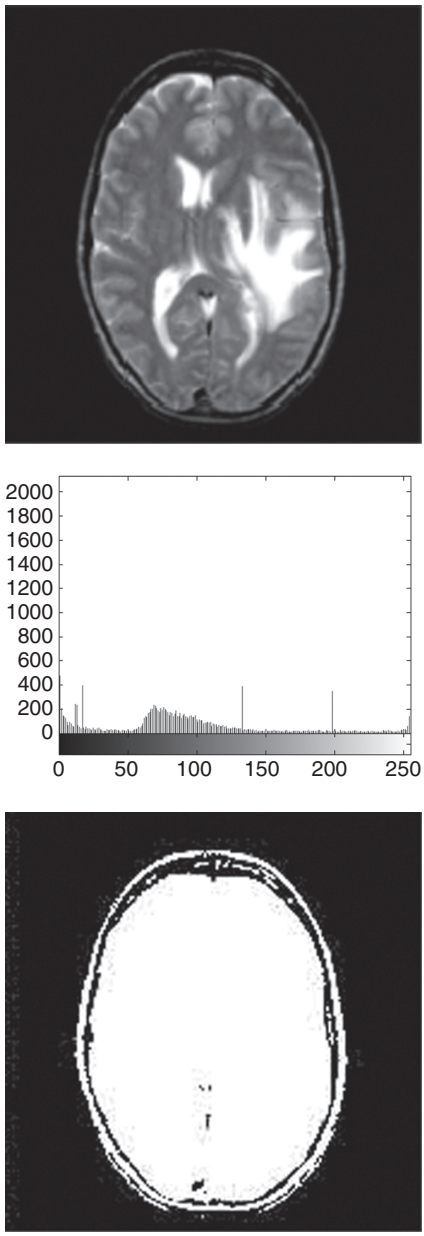


Figure 10.3 The original MR brain image (top), its gray-level histogram (middle), and the segmented image (bottom) using a gray-value threshold $T = 12$ at the first major valley point in the histogram.

A threshold between the values 12 and 254 can be determined interactively to segment the regions of interest. An alternative approach is to recompute another histogram for the gray values between 12 and 255 (corresponding to the segmented brain area only) and then to analyze this histogram for peaks and values to determine

Figure 10.4 Two segmented MR brain images using a gray-value threshold $T = 166$ (top) and $T = 225$ (bottom).

additional thresholds recursively (5, 7). Figure 10.4 shows two segmentations obtained using the thresholds at, respectively, gray values 166 and 225. These thresholds were determined by analyzing the peaks and valleys of the remaining histograms in a recursive manner. The brain regions related to the white matter, cerebrospinal fluid in the sulci, ventricles, and a lesion (in the right half of the image) can be seen in the segmented image obtained using the gray-value threshold of 166. The segmented image using the gray-value threshold of 255 shows only the ventricles and lesion. The higher gray values of the ventricle and lesion regions are due to higher proton density and T_2 relaxation times of MR imaging parameters.

10.2.1 Optimal Global Thresholding

To determine an optimal global gray-value threshold for image segmentation, parametric distribution-based methods can be applied to the histogram of an image (2, 13). Let us assume that the histogram of an image to be segmented has two Gaussian distributions belonging to two respective classes, such as background and object. Thus, the histogram can be represented by a mixture probability density function $p(z)$ as

$$p(z) = P_1 p_1(z) + P_2 p_2(z) \tag{10.10}$$

where $p_1(z)$ and $p_2(z)$ are the Gaussian distributions of classes 1 and 2, respectively, with the class probabilities of P_1 and P_2 such that

$$P_1 + P_2 = 1. \tag{10.11}$$

Using a gray-value threshold T, a pixel in the image $f(x, y)$ can be sorted into class 1 or class 2 in the segmented image $g(x, y)$ as

$$g(x, y) = \begin{cases} \text{Class 1} & \text{if } f(x,y) > T \\ \text{Class 2} & \text{if } f(x,y) \le T \end{cases}. \tag{10.12}$$

Let us define the error probabilities of misclassifying a pixel as

$$E_1(T) = \int_{-\infty}^{T} p_2(z)dz$$

and

$$E_2(T) = \int_{-\infty}^{T} p_1(z)dz \tag{10.13}$$

where $E_1(T)$ and $E_2(T)$ are, respectively, the probability of erroneously classifying a class 1 pixel to class 2 and a class 2 pixel to class 1.

The overall probability of error in pixel classification using the threshold T is then expressed as

$$E(T) = P_2(T)E_1(T) + P_1(T)E_2(T). \tag{10.14}$$

For image segmentation, the objective is to find an optimal threshold T that minimizes the overall probability of error in pixel classification. The optimization process requires the parameterization of the probability density distributions and likelihood of both classes. These parameters can be determined from a model or set of training images (10, 13).

Let us assume σ_i and μ_i to be the standard deviation and mean of the Gaussian probability density function of the class i (i = 1, 2 for two classes) such that

$$p(z) = \frac{P_1}{\sqrt{2\pi}\sigma_1} e^{-(z-\mu_1)^2/2\sigma_1^2} + \frac{P_2}{\sqrt{2\pi}\sigma_2} e^{-(z-\mu_2)^2/2\sigma_2^2}. \tag{10.15}$$

The optimal global threshold T can be determined by finding a general solution that minimizes Equation 10.14 with the mixture distribution in Equation 10.15 and thus satisfies the following quadratic expression (2)

$$AT^2 + BT + C = 0$$

where

$$\begin{aligned} A &= \sigma_1^2 - \sigma_2^2 \\ B &= 2\left(\mu_1\sigma_2^2 - \mu_2\sigma_1^2\right) \\ C &= \sigma_1^2\mu_2^2 - \sigma_2^2\mu_1^2 + 2\sigma_1^2\sigma_2^2 \ln\left(\sigma_2 P_1 / \sigma_1 P_2\right). \end{aligned} \tag{10.16}$$

If the variances of both classes can be assumed to ¾ equal to σ^2, the optimal threshold T can be determined as

$$T = \frac{\mu_1 + \mu_2}{2} + \frac{\sigma^2}{\mu_1 - \mu_2} \ln\left(\frac{P_2}{P_1}\right). \tag{10.17}$$

It should be noted that in case of equal likelihood of classes, the above expression for determining the optimal threshold is simply reduced to the average of the mean values of two classes.

10.2.2 Pixel Classification Through Clustering

In the histogram-based pixel classification method for image segmentation, the gray values are partitioned into two or more clusters, depending on the peaks in the histogram to obtain thresholds. The basic concept of segmentation by pixel classification can be extended to clustering the gray values or feature vectors of pixels in the image. This approach is particularly useful when images with pixels representing a feature vector consisting of multiple parameters of interest are to be segmented. For example, a feature vector may consist of gray value, contrast, and local texture measures for each pixel in the image. A color image may have additional color components in a specific representation such as red, green, and blue components in the RGB color coordinate system that can be added to the feature vector. MR or multimodality medical images may also require segmentation using a multidimensional feature space with multiple parameters of interest.

Images can be segmented by pixel classification through clustering of all features of interest. The number of clusters in the multidimensional feature space thus represents the number of classes in the image. As the image is sorted into cluster classes, segmented regions are obtained by checking the neighborhood pixels for the same class label. However, clustering may produce disjointed regions with holes or regions with a single pixel. After the image data are clustered and pixels are classified, a postprocessing algorithm such as region growing, pixel connectivity, or a rule-based algorithm is usually applied to obtain the final segmented regions (19). In the literature there are a number of algorithms for clustering used for a wide range of applications (13, 18, 21, 23–27).

10.2.2.1 Data Clustering Clustering is the process of grouping data points with similar feature vectors together in a single cluster while data points with dissimilar feature vectors are placed in different clusters. Thus, the data points that are close to each other in the feature space are clustered together. The similarity of feature vectors can be represented by an appropriate distance measure such as Euclidean or Mahalanobis distance (30). Each cluster is represented by its mean (centroid) and variance (spread) associated with the distribution of the corresponding feature vectors of the data points in the cluster. The formation of clusters is optimized with respect to an objective function involving prespecified distance and similarity measures, along with additional constraints such as smoothness.

Since similarity is fundamental to the formation of a cluster, a measure of the similarity between two patterns drawn from the same feature space is essential (30).

For image segmentation, the feature space can simply be represented by gray levels of pixels. However, other features such as contrast can be added to the feature space. It is common to compute dissimilarity between two patterns using a distance measure defined in a feature space.

The Minkowski distance $d_p(\mathbf{x}, \mathbf{y})$ of order p (also known as p-norm distance) defines a distance between two vectors $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ and $\mathbf{y} = (y_1, y_2, \ldots, y_n)$ as

$$d_p(\mathbf{x}, \mathbf{y}) = \left(\sum_{i=1}^{n} |x_i - y_i|^p \right)^{1/p}. \tag{10.18}$$

Euclidean distance $d_2(\mathbf{x}, \mathbf{y})$ is the most popular metric; it is defined as the 2-norm distance measure as:

$$d_2(\mathbf{x}, \mathbf{y}) = \left(\sum_{i=1}^{n} |x_i - y_i|^2 \right)^{1/2} = \|\mathbf{x} - \mathbf{y}\|_2 . \tag{10.19}$$

The Euclidean distance has an intuitive appeal as it is commonly used to evaluate the proximity of objects in 2- or 3-D space. It works well when a data set has "compact" or "isolated" clusters (30, 31). The Minkowski metric favors the largest scaled feature, which dominates others. The problem can be addressed by proper normalization or other weighting schemes applied in the feature space. Linear correlation among features can also distort distance measures. This distortion can be alleviated by applying a whitening transformation to the data by using the Mahalanobis distance measure $d_M(\mathbf{x}, \mathbf{y})$ as

$$d_M(\mathbf{x}, \mathbf{y}) = \left((\mathbf{x} - \mathbf{y}) A^{-1} (\mathbf{x} - \mathbf{y})^T \right)^{1/2} \tag{10.20}$$

where A is the covariance matrix.

In this process, $d_M(\mathbf{x}, \mathbf{y})$ assigns different weights to different features based on their variances and correlations. It is implicitly assumed here that both vectors have the same statistical distributions.

Conventional data clustering methods can be classified into two broad categories: hierarchical and partitional. In hierarchical clustering, the number of clusters need not be specified a priori. Hierarchical clustering methods consider only local neighbors in each step. For this reason, it is difficult to handle overlapping clusters through the hierarchical clustering method. In addition, hierarchical clustering is static in the sense that data points allocated to a cluster in the early stages may not be moved to a different cluster.

Partitional clustering methods develop a clustering structure by optimizing a criterion function defined either locally (on a subset of the patterns) or globally (defined over all of the data). Partitional clustering can be further divided into two classes: crisp clustering and fuzzy clustering. In crisp clustering, a data point belongs to only one cluster and clusters are separated by crisp boundaries among them. In fuzzy clustering methods, data points belong to all clusters through a degree determined by the membership function (32, 33). Partitional algorithms are dynamic in the sense that data points can be efficiently moved from one cluster to another. They can incorporate knowledge about the shape or size of clusters by using appropriate prototypes and constraints.

A popular hierarchical clustering algorithm is the agglomerative method, which is simply based on the distance between clusters. From a representation of single-point-based clusters, two clusters that are nearest and satisfy a similarity criterion are progressively merged to form a reduced number of clusters. This is repeated until just one cluster is obtained, containing the entire data set. Let us suppose that there are n points in the data vector $\mathbf{x} = (x_1, x_2, \ldots, x_n)$; the initial number of clusters will then be equal to n. An agglomerative algorithm for clustering can be defined as follows.

algorithm (agglomerative hierarchical clustering):

Step 1: for $i = 1, \ldots, n$ define cluster $C_i = \{x_i\}$

Step 2: Loop: While there is more than one cluster left compute distance between any two clusters find cluster C_j with minimum distance to C_i
$C_i = C_i \cup C_j$;
Remove cluster C_j;

End

In the above algorithm, a distance measure should be carefully chosen. Euclidean distance measure is commonly used but other measures can also be used depending on the data. It can be noted from the above algorithm that the agglomerative clustering method yields a hierarchical structure of clusters that can be shown graphically in a dendrogram showing the merging links among the data points. A dendrogram shows all data points as individual clusters at the bottom with progressively merging links and reducing number of clusters until all data points form one single cluster at the top.

A nonhierarchical or partitional clustering algorithm provides clusters as a result of the optimization of data partitioning in the feature space. Clusters can be obtained with crisp partitions (as done in k-means clustering method) or fuzzy partitions (as done in fuzzy c-means clustering method). It should be noted that these algorithms are described here for use in image segmentation but are also used for feature analysis and classification as described in Chapter 11.

10.2.2.2 k-Means Clustering The k-means clustering method is a popular approach to partition d-dimensional data into k clusters such that an objective function providing the desired properties of the distribution of feature vectors of clusters in terms of similarity and distance measures is optimized (31). A generalized k-means clustering algorithm initially places k clusters at arbitrarily selected cluster centroids $\mathbf{v_i}$; $i = 1, \ldots, 2, k$ and modifies centroids for the formation of new cluster shapes, optimizing the objective function. The k-means clustering algorithm includes the following steps:

1. Select the number of clusters k with initial cluster centroids $\mathbf{v_i}$; $i = 1, \ldots 2, k$.
2. Partition the input data points into k clusters by assigning each data point $\mathbf{x}_j$ to the closest cluster centroid $\mathbf{v_i}$ using the selected distance measure, e.g., Euclidean distance, defined as

$$d_{ij} = \|\mathbf{x}_j - \mathbf{v}_i\| \tag{10.21}$$

where $\mathbf{X} = \{\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x}_n\}$ is the input data set.

3. Compute a cluster assignment matrix **U** representing the partition of the data points with the binary membership value of the *j*th data point to the *i*th cluster such that

$$\mathbf{U} = \left[u_{ij}\right] \tag{10.22}$$

where $u_{ij} \in \{0, 1\}$ for all i, j

$$\sum_{i=1}^{k} u_{ij} = 1 \quad \text{for all } j \quad \text{and} \quad 0 < \sum_{j=1}^{n} u_{ij} < n \quad \text{for all } i$$

4. Recompute the centroids using the membership values as

$$\mathbf{v_i} = \frac{\sum_{j=1}^{n} u_{ij}\mathbf{x_j}}{\sum_{j=1}^{n} u_{ij}} \text{ for all } i \tag{10.23}$$

5. If cluster centroids or the assignment matrix does not change from the previous iteration, stop; otherwise go to step 2.

The *k*-means clustering method optimizes the sum-of-squared-error-based objective function $J_w(\mathbf{U}, \mathbf{v})$ such that

$$J_w(\mathbf{U}, \mathbf{v}) = \sum_{i=1}^{k}\sum_{j=1}^{n} \left\|\mathbf{x}_j - \mathbf{v_i}\right\|^2 \tag{10.24}$$

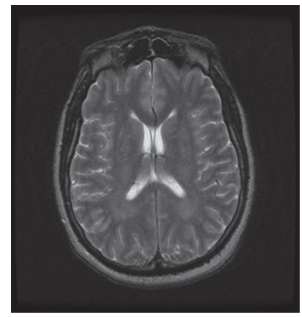
It can be noted from the above algorithm that the *k*-means clustering method is quite sensitive to the initial cluster assignment and the choice of the distance measure. Additional criterion such as within-cluster and between-cluster variances can be included in the objective function as constraints to force the algorithm to adapt the number of clusters *k* (as needed for optimization of the objective function). Figure 10.5 shows the results of *k*-means clustering on a T_2-weighted MR brain image with $k = 9$. Regions segmented from different clusters are shown in Figure 10.5b.

10.2.2.3 Fuzzy c-Means Clustering The *k*-means clustering method utilizes hard binary values for the membership of a data point to the cluster. The fuzzy *c*-means (FCM) clustering method utilizes an adaptable membership value that can be updated based on the distribution statistics of the data points assigned to the cluster minimizing the following objective function $J_m(\mathbf{U}, \mathbf{v})$

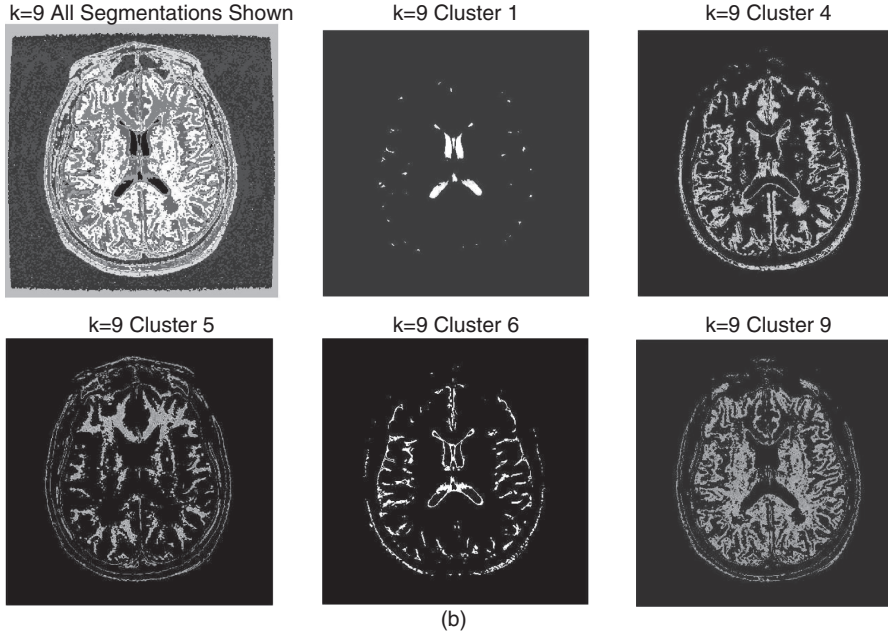
$$J_m(\mathbf{U}, \mathbf{v}) = \sum_{i=1}^{c}\sum_{j=1}^{n} u_{ij}^m d_{ij}^2 = \sum_{i=1}^{c}\sum_{j=1}^{n} u_{ij}^m \left\|\mathbf{x_j} - \mathbf{v_i}\right\|^2 \tag{10.25}$$

where *c* is the number of clusters, *n* is the number of data vectors, u_{ij} is the fuzzy membership, and *m* is the fuzziness index.

Based on the constraints defined in the distribution statistics of the data points in the clusters, the fuzziness index can be defined between 1 and a very large value for the highest level of fuzziness (maximum allowable variance within a cluster). The membership values in the FCM algorithm can be defined as (33):



(a)



(b)

Figure 10.5 (a) A T_2-weighted MR brain image used for segmentation. (b) Results of segmentation of the image shown in Figure 2a using k-means clustering algorithm with $k = 9$: Top left: all segmented regions belonging to all nine clusters. Top middle: regions segmented from cluster $k = 1$. Top right: regions segmented from cluster $k = 4$. Bottom left: regions segmented from cluster $k = 5$. Bottom middle: regions segmented from cluster $k = 6$. Bottom right: regions segmented from cluster $k = 9$. (Courtesy of Don Adams, Arwa Gheith, and Valerie Rafalko from their class project).

$$0 \le u_{ij} \le 1 \quad \text{for all } i, j$$
$$\sum_{i=1}^{c} u_{ij} = 1 \quad \text{for all } j \quad \text{and} \quad 0 < \sum_{j=1}^{n} u_{ij} < n \quad \text{for all } i. \tag{10.26}$$

The algorithm described for k-means clustering can be used for fuzzy c-means clustering with the update of the fuzzy membership values as defined in Equation 10.26, minimizing the objective function as defined in Equation 10.25.

10.2.2.4 An Adaptive FCM Algorithm Additional constraints such as smoothness can be included in the objective function for clustering for better results. An adaptive version of the fuzzy *c*-means (FCM) clustering algorithm using a weighted derivative information is described by Pham and Prince (34). As the objective function is minimized for clustering, the minimization of the derivative information provides smooth homogenous clusters. The objective function $J_{AFCM}(\mathbf{U}, \mathbf{v})$ to be minimized for an adaptive fuzzy *c*-means clustering algorithm can be given as

$$J_{ACFM}(\mathbf{U}, \mathbf{v}) = \sum_{i=1}^{c}\sum_{j=1}^{n} u_{ij}^{m} \left\|\mathbf{x_j} - g_j \mathbf{v_i}\right\|^2 + \lambda_1 \sum_{j=1}^{n}\sum_{r=1}^{R} (D_r \otimes g)_j^2 + \lambda_2 \sum_{j=1}^{n}\sum_{r=1}^{R}\sum_{s=1}^{R} (D_r \otimes D_s \otimes g)_j^2 \tag{10.27}$$

where g_j is the unknown gain field to be estimated during the iterative clustering process, m is the fuzziness index ($m > 1$), R is the dimension of the image, D_r is the derivative operator along the rth dimension of the image, and λ_1 and λ_2 are, respectively, the weights for the first-order and second-order derivative terms.

Thus, the second and third terms in the objective function provide the smoothness constraints on clustering.

The adaptive fuzzy *c*-means algorithm involves the following steps (34):

1. Start with initial centroids $\mathbf{v_i}$; $i = 1, \ldots 2, c$ for c clusters and set the initial gain field $g_j = 1$ for all values of j belonging to the image data.
2. Compute membership values as

$$u_{ij} = \frac{\left\|\mathbf{x_j} - g_j \mathbf{v_i}\right\|^{-2/(m-1)}}{\sum_{i=1}^{c} \left\|\mathbf{x_j} - g_j \mathbf{v_i}\right\|^{-2/(m-1)}}. \tag{10.28}$$

3. Compute new centorids as

$$\mathbf{v_i} = \frac{\sum_{j=1}^{n} u_{ij}^{m} g_j \mathbf{x_j}}{\sum_{j=1}^{n} u_{ij}^{m} g_j^2}, \quad \text{for all } i = 1, 2, \ldots, c. \tag{10.29}$$

4. Compute a new gain field by solving the following equation for g_j

$$\sum_{i=1}^{c} u_{ij}^{m}(\mathbf{x_j}, \mathbf{v_i}) = g_j \sum_{i=1}^{c} u_{ij}^{m}(\mathbf{v_i}, \mathbf{v_i}) + \lambda_1 (H_1 \otimes g)_j + \lambda_2 (H_2 \otimes g)_j$$

with

$$H_1 = \sum_{r=1}^{R} (D_r \otimes \bar{D}_r)_j$$

$$H_2 = \sum_{r=1}^{R} ((D_r \otimes D_r) \otimes (\bar{D}_s \otimes \bar{D}_s)_j \tag{10.30}$$

where $\bar{D}_r$ is the mirror reflection of D_r derivative operator.

5. If there is no difference in the centroids from the previous iteration, or if there is not a significant difference in the membership values, the algorithm has converged. Stop. Otherwise, go to step 2.

The adaptive FCM algorithm provides good regularization in the clustering process through the first- and second-order derivative terms that force the bias field to be smooth. For large values of weights, λ_1 and λ_2, the algorithm yields a constant bias field leading to the standard FCM method.

10.3. REGION-BASED SEGMENTATION

Region-growing based segmentation algorithms examine pixels in the neighborhood based on a predefined similarity criterion. The neighborhood pixels with similar properties are merged to form closed regions for segmentation. The region-growing approach can be extended to merging regions instead of merging pixels to form larger meaningful regions with similar properties. Such a region-merging approach is quite effective when the original image is segmented into a large number of regions in the preprocessing phase. Large meaningful regions may provide better correspondence and matching to the object models for recognition and interpretation. An alternate approach is region splitting, in which either the entire image or large regions of the image are split into two or more regions based on a heterogeneity or dissimilarity criterion. For example, if a region has a bimodal distribution of gray-value histogram, it can be split into two regions of connected pixels with gray values falling into their respective distributions. The basic difference between region- and thresholding-based segmentation approaches is that region-growing methods guarantee the segmented regions of connected pixels. On the other hand, pixel thresholding-based segmentation methods as defined in the previous section may yield regions with holes and disconnected pixels.

10.3.1 Region-Growing

Region-growing methods merge pixels of similar properties by examining the neighborhood pixels. The process of merging pixels continues, with the region adapting a new shape and size, until there are no eligible neighborhood pixels to be added to the current region. Thus, the region-growing process requires two criteria: a similarity criterion that defines the basis for inclusion of pixels in the growth of the region, and a stopping criterion that stops the growth of the region. The stopping criterion is usually based on the minimum number or percentage of neighborhood pixels required to satisfy the similarity criterion for inclusion in the growth of the region.

An example of a region-growing process is shown in Figure 10.6. Let us assume that a pixel at the center, as shown in Figure 10.6, is the origin of the region-growing process. In the first iteration, there are 8 pixels in the 3×3 neighborhood of the center pixel satisfying the similarity criterion. All of these pixels are grouped tighter in the region, which has now grown to a 3×3 size with a square shape. In the second iteration, the pixels in the 5×5 neighborhood are examined for similarity

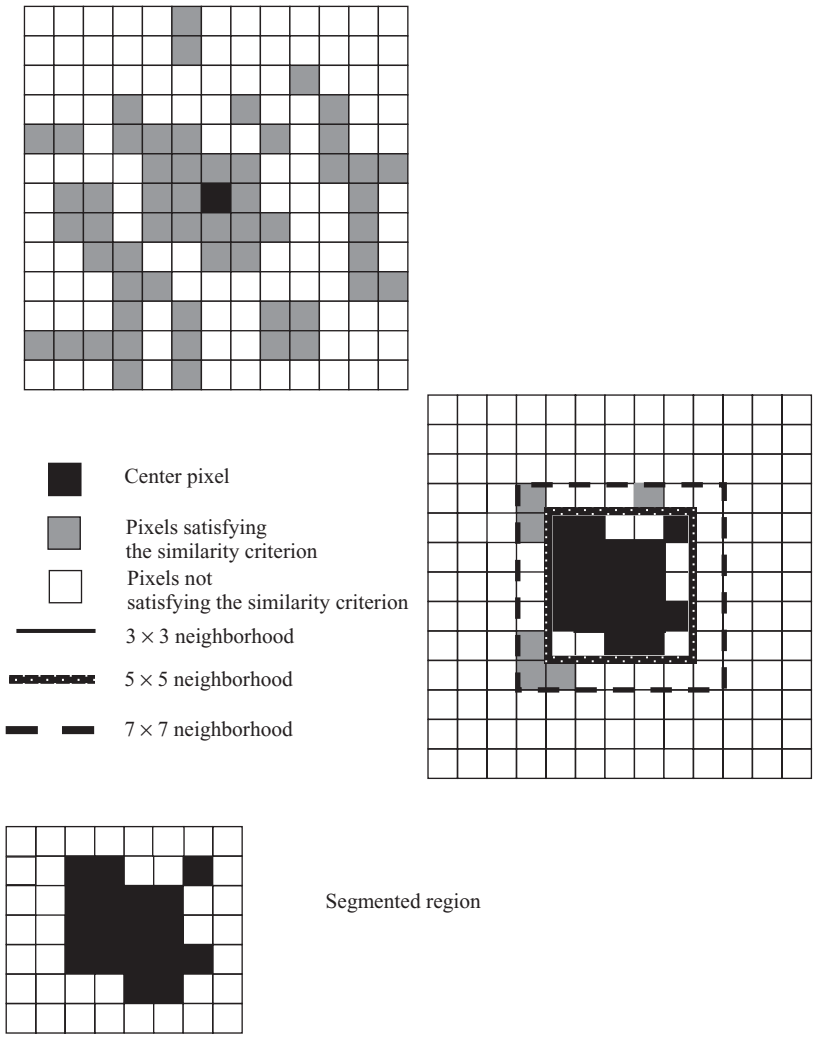


Figure 10.6 A pixel map of an image (top) with the region-growing process (middle) and the segmented region (bottom).

criterion. There are 16 new pixels in the 5 × 5 neighborhood, out of which 9 spixels satisfy the similarity criterion. These pixels are now included in the region. Let us assume that the stopping criterion is defined as the minimum percentage of pixels to be included in the growth of the region to be 30%. Therefore, in the second iteration, 56% of neighborhood pixels (9/16) are included in the growth of the region. In the third iteration, only 25% of the new neighborhood pixels satisfy the similarity criterion. The region-growing process stops when the stopping criterion is met. The region thus grown is shown in Figure 10.6. In this example, the neighborhoods are grown in a fixed shape, that is, 3 × 3 to 5 × 5 to 7 × 7, and so on. A feature-adaptive region-growing method is described in Chapter 9 for feature enhancement, which

can be used to grow regions that are adaptive to the size and shape of the growing region by examining the neighborhood (9, 10).

In the region-merging algorithms, an image may be partitioned into a large number of potential homogeneous regions. For example, an image of 1024×1024 pixels can be portioned into regions of 8×8 pixels. Each region of 8×8 pixels can now be examined for homogeneity of predefined property such as gray values, contrast, and texture. If the histogram of the predefined property for the region is unimodal, the region is said to be homogeneous. Two neighborhood regions can be merged if they are homogeneous and satisfy a predefined similarity criterion. The similarity criterion imposes constraints on the value of the property with respect to its mean and variance values. For example, two homogeneous regions can be merged if the difference between their mean gray values is within 10% of the entire dynamic range and the difference between their variances is within 10% of the variance in the image. These thresholds may be selected heuristically or through probabilistic models (10, 13). It is interesting to note that the above criterion can be easily implemented as a conditional rule in a knowledge-based system. Region-merging or region-splitting (described in the next section) methods have been implemented using a rule-based system for image segmentation (23).

Model-based systems typically encode knowledge of anatomy and image acquisition parameters. Anatomical knowledge can be modeled symbolically, describing the properties and relationships of individual structures; geometrically, either as masks or templates of anatomy; or by using an atlas (16, 17, 23, 24).

Figure 10.7 shows an MR brain image and the segmented regions for ventricles. The knowledge of the anatomical locations of ventricles was used to establish the initial seed points for region-growing. A feature-adaptive region-growing method was used for segmentation.

10.3.2 Region-Splitting

Region-splitting methods examine the heterogeneity of a predefined property of the entire region in terms of its distribution and the mean, variance, minimum, and maximum values. If the region is evaluated as heterogeneous, that is, it fails the similarity or homogeneity criterion, the original region is split into two or more regions. The region-splitting process continues until all regions satisfy the homogeneity criterion individually.

In the region-splitting process, the original region R is split into $R_1, R_2, \ldots, R_n$ subregions such that the following conditions are met (2):

1. Each region, R_i; $i = 1, 2, \ldots, n$ is connected.
2. $\bigcup_{i=1}^{n} R_i = R$
3. $R_i \bigcap R_j = \mathrm{O}$ for all i, j; $i \neq j$
4. $H(R_i) = \mathrm{TRUE}$ for $i = 1, 2, \ldots, n$.
5. $H(R_i \bigcup R_j) = \mathrm{FALSE}$ for $i \neq j$

where $H(R_i)$ is a logical predicate for the homogeneity criterion on the region R_i.

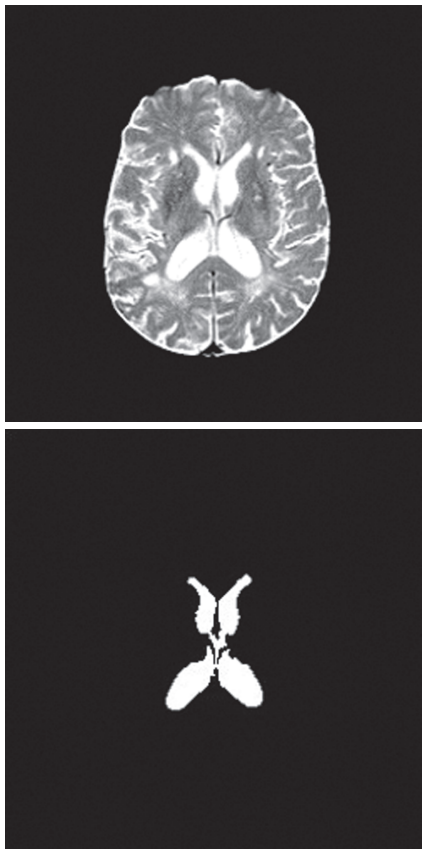


Figure 10.7 A T_2-weighted MR brain image (top) and the segmented ventricles (bottom) using the region-growing method.

Region-splitting methods can also be implemented by rule-based systems and quad trees. In the quad tree-based region-splitting method, the image is partitioned into four regions that are represented by nodes in a quad tree. Each region is checked for the homogeneity and evaluated for the logical predicate $H(R_i)$. If the region is homogeneous, no further action is taken for the respective node. If the region is not homogeneous, it is further split into four regions. Figure 10.8 shows a quad-tree structure. The image was initially split into four regions, R_1, R_2, R_3, and R_4. Regions R_1 and R_3 were found to be homogeneous and therefore no further split was needed. Regions R_2 and R_4 failed the homogeneous predicate test and were therefore further split into four regions respectively.

10.4. ADVANCED SEGMENTATION METHODS

The problem of segmenting medical images into anatomically and pathologically meaningful regions has been addressed using various approaches, including model-based estimation methods and rule-based systems (11, 15–25). Nevertheless,

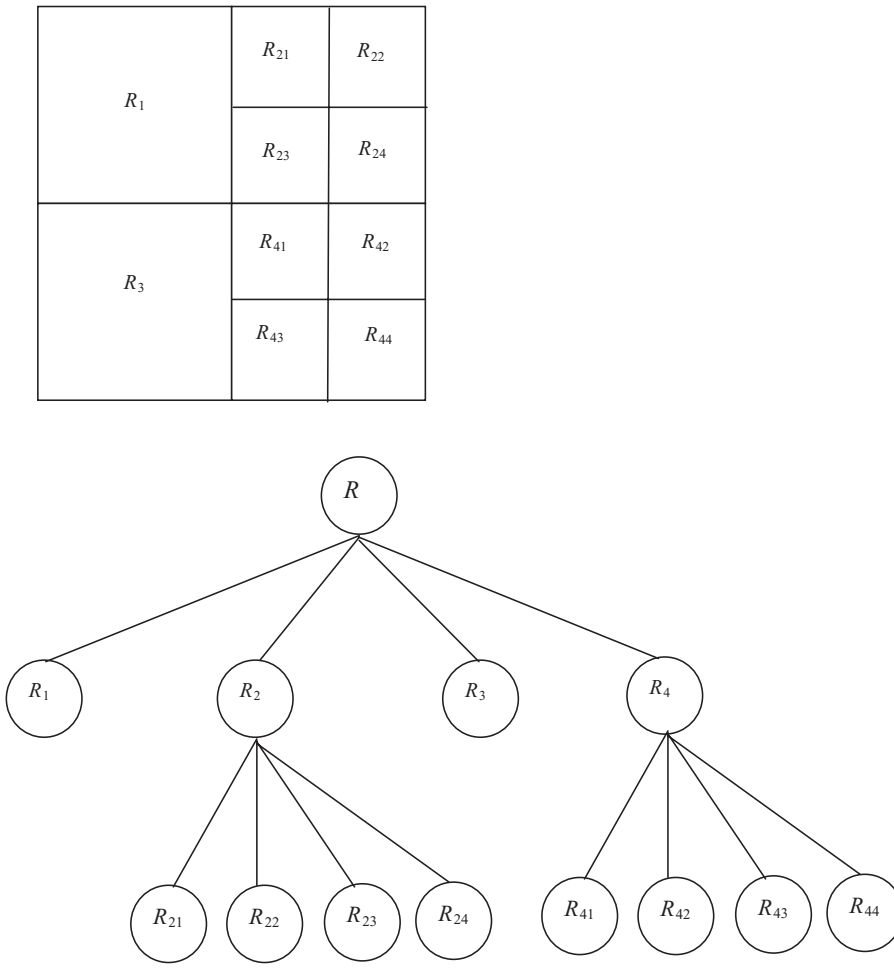


Figure 10.8 An image with quad region-splitting process (top) and the corresponding quad-tree structure (bottom).

automatic (or semi-automatic with minimal operator interaction) segmentation methods for specific applications are still current topics of research. This is due to the great variability in anatomical structures and the challenge of finding a reliable, accurate, and diagnostically useful segmentation.

A rule-based low-level segmentation system for automatic identification of brain structures from MR images has been described by Raya (16). Neural network-based classification approaches have also been applied for medical image segmentation (10, 26, 27).

10.4.1 Estimation-Model Based Adaptive Segmentation

A multilevel adaptive segmentation (MAS) method is used to segment and classify multiparameter MR brain images into a large number of classes of physiological

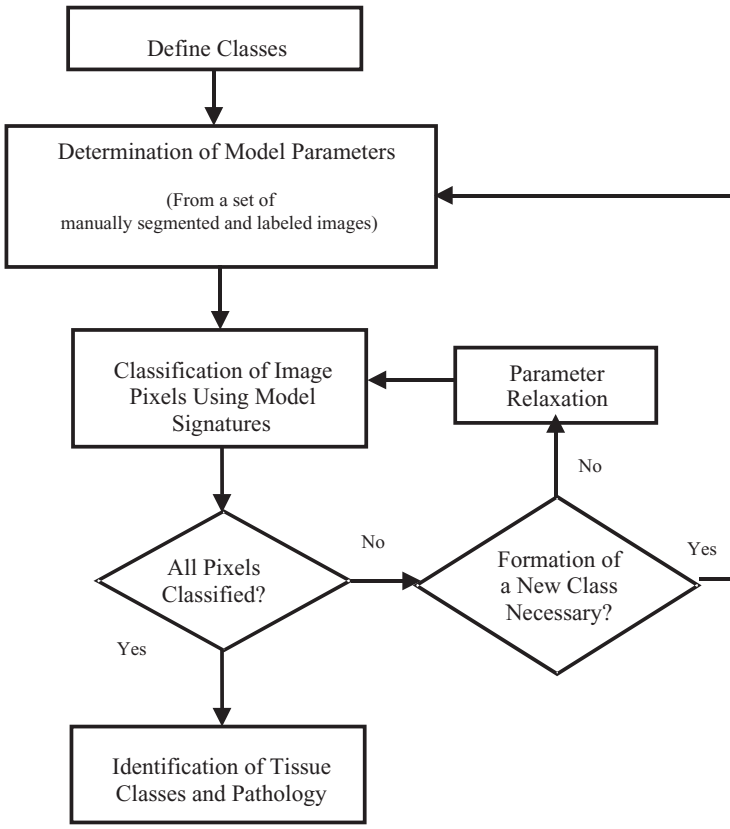


Figure 10.9 The overall approach of the MAS method.

and pathological interest (22). The MAS method is based on an estimation of signatures for each segmentation class for pixel-by-pixel classification. It can also create new classes, if necessary, when there is a significant discrepancy between the measured parameters of the image and the model signatures. The MAS method consists of a five-step procedure for generating brain model signatures, as illustrated in Figure 10.9 (22).

Let us represent a multiparameter image to be segmented with a pair of random variables

$$\{C_{m,n}, X_{m,n}\} \tag{10.31}$$

where $C_{m,n}$ is the class of the pixel (m, n) representing the spatial variability of the brain classes, and can take the values in the discrete set $\{1, 2, \ldots, K\}$ representing a specific class, and $X_{m,n}$ is a d-dimensional random variable representing the data vector of pixel (m, n) with dimension d that is based on the number of imaging parameters, such as T_1, T_2, proton density, and $G_d + T_1$.

The required parameters to develop an estimation model can be obtained using a priori knowledge or manual segmentation on a set of training images. Let us

assume an ensemble of brain images manually classified; the parameters were estimated in the following way. The ith class mean vectors, $\hat{\mu}_i$, is

$$\hat{\mu}_i = \frac{1}{n}\sum_j x_j \tag{10.32}$$

where n is the number of pixels in the ith class, and x_j is the jth of n multidimensional vectors that comprise the class. The dimension of x_j corresponds to the number of image modalities used in the analysis. The covariance matrix of class i, $\hat{\Sigma}_i$, is

$$\hat{\Sigma}_i = \frac{1}{n-1}\sum_j (x_j - \hat{\mu}_i)(x_j - \hat{\mu}_i)^t \tag{10.33}$$

The summations in Equations 10.29 and 10.30 are expressed over all the pixels belonging to the ith class.

Given that $C_{m,n} = i$, the distribution of $X_{m,n}$ is estimated to satisfy the multivariate normal distribution described by the density function

$$\hat{p}_i(x) = \frac{1}{(2\pi)^{d/2}\left|\hat{\Sigma}_i\right|^{1/2}} \exp\left[\frac{-(x-\hat{\mu}_i)}{2\hat{\Sigma}_i(x-\hat{\mu}_i)}\right] \tag{10.34}$$

where x is a d-element column vector, $\hat{\mu}_i$ is a d-element estimated mean vector for the class i calculated from Equation 10.29, $\hat{\Sigma}_i$ is the estimated $d \times d$ covariance matrix for class i calculated from Equation 10.30, and d is the dimension of multiparameter data.

Four transition matrices $P_r(m, n) = [p_{ijr}(m, n)]$ can be estimated, where r is a direction index (with four spatial connectedness directions) and $p_{ijr}(m, n)$ are the transition probabilities defined by

$$\begin{aligned}
p_{ij1}(m,n) &= P\{C_{m,n} = j \mid C_{m,n-1} = i\} \\
p_{ij2}(m,n) &= P\{C_{m,n} = j \mid C_{m+1,n} = i\} \\
p_{ij3}(m,n) &= P\{C_{m,n} = j \mid C_{m,n+1} = i\} \\
p_{ij4}(m,n) &= P\{C_{m,n} = j \mid C_{m-1,n} = i\}.
\end{aligned} \tag{10.35}$$

Transition probabilities, in general, can be given by

$$p_{ij1}(m,n) = \frac{\sum_b \{\text{pix} | C_{m,n} = j, C_{m,n-1} = i\}}{\sum_b \{\text{pix} | C_{m,n-1} = i\}} \tag{10.36}$$

where $\sum_b\{\text{pix} \mid P\}$ denotes the number of pixels with the property P in the ensemble of brains used to generate the model.

Due to the relatively small number of brain images in the training set, averaging over a small neighborhood of pixels around the pixel of interest (m, n) is performed. Thus, the estimate can be expressed as

$$p_{ij1}(m,n) = \frac{\sum_b \sum_n \{\text{pix} | C_{m,n} = j, C_{m,n-1} = i\}}{\sum_b \sum_n \{\text{pix} | C_{m,n-1} = i\}} \tag{10.37}$$

where $\sum_n$ indicates the averaging over the neighborhood.

The equilibrium transition probabilities are estimated using a similar procedure and are

$$\pi_i(mn) = \frac{\sum_b \sum_n \{\text{pix} | C_{m,n} = i\}}{\sum_b \sum_n \{\text{pix}\}}. \tag{10.38}$$

To perform maximum likelihood discriminant analysis, the class random variable $C_{m,n}$ is assumed to constitute a K-state Markov random field. Rows and columns of $C_{m,n}$ constitute segments of K-state Markov chains. Chains are specified by the $K \times K$ transition matrix $P = [p_{ij}]$ where

$$p_{ij} = P\{C_{m,n} = j \mid C_{m,n-1} = i\} \tag{10.39}$$

with the equilibrium probabilities ($\pi_1, \pi_2, \ldots, \pi_K$).

Thus, the probability that a pixel belongs to a specific class i is

$$P\{C_{m,n} = i \mid x_{kl}, (k,l) \in N(m,n)\} \tag{10.40}$$

where $N(m, n)$ is some neighborhood of the pixel (m, n).

The conditional probabilities can now be expressed as

$$P\{C_{m,n} = i \mid x_{kl}, (k,l) \in N(m,n)\} = \frac{P\{C_{m,n} = i, X_{m,n} \mid X_{m\pm1,n}, X_{m,n\pm1}\}}{P\{X_{m,n} \mid X_{m\pm1,n}, X_{m,n\pm1}\}}$$

and

$$P\{C_{m,n} = i \mid x_{kl}, (k,l) \in N(m,n)\} = \frac{P\{X_{m,n} \mid C_{m,n} = i, X_{m\pm1,n}, X_{m,n\pm1}\} P\{C_{m,n} = i \mid X_{m\pm1,n}, X_{m,n\pm1}\}}{P\{X_{m,n} \mid X_{m\pm1,n}, X_{m,n\pm1}\}} \tag{10.41}$$

where

$$P\{\circ \mid X_{m\pm1,n}, X_{m,n\pm1}\} \equiv P\{\circ \mid X_{m-1,n}\} P\{\circ \mid X_{m,n-1}\} P\{\circ \mid X_{m+1,n}\} P\{\circ \mid X_{m,n+1}\} \tag{10.42}$$

Using class conditional independence of x_{mn} and applying Bayes' rule, Equation 10.38 can be rewritten as

$$P\{C_{m,n} = i \mid x_{kl}, (k,l) \in N(m,n)\} = \frac{P\{X_{m,n} \mid C_{m,n} = i\} P\{X_{m\pm1,n}, X_{m,n\pm1} \mid C_{m,n} = i\} P\{C_{m,n} = i\}}{P\{X_{m,n} \mid X_{m\pm1,n}, X_{m,n\pm1}\} P\{X_{m\pm1,n}, X_{m,n\pm1}\}} \tag{10.43}$$

with

$$P\{X_{m-1,n} \mid C_{m,n} = i\} = \sum_{j=1}^{N} P\{X_{m-1,n} \mid C_{m,n} = j\} P\{C_{m-1,n} = j \mid C_{m,n} = i\} \equiv H_{\boldsymbol{m}-1,n}(i). \tag{10.44}$$

With substitutions, the probability that the current pixel, mn, belongs to class i given the characteristics of the pixels in the neighborhood of mn is expressed as

$$P\{C_{m,n} = i \mid x_{kl}, (k,l) \in N(m,n)\} = \frac{P\{C_{m,n} = i \mid X_{m,n}\} P\{X_{m,n}\} H_{m-1,n}(i) H_{\boldsymbol{m},n-1}(i) H_{m+1,n}(i) H_{m,n+1}(i)}{P\{X_{m,n} \mid X_{m\pm1,n}, X_{m,n\pm1}\} P\{X_{m\pm1,n}, X_{m,n\pm1}\}}. \tag{10.45}$$

The brain model consists of class probabilities, transition probabilities, mean vectors, and covariance matrices of class clusters in d dimensions (T_1, T_2, proton density, $G_d + T_1$, and perfusion) space. After the model is developed, an image to be segmented is automatically classified on a pixel-by-pixel basis using the model signatures. The multiparameter measurement vector of each pixel in the image to be classified is compared with the model signature vector of each brain class. A pixel is classified in a particular class if its distance from the model signature is less than some predefined threshold. Thus, pixel classification is done using the distances in the corresponding multidimensional metric space from the model class signatures. The thresholds here are selected in such a way as to classify pixels with relatively pure content of the corresponding class. This step provides a successful classification of a limited number of pixels that are well inside the spatial extents of brain tissue regions.

The next level of processing deals with data on the neighborhood level. For the remaining unclassified pixels, a spatial neighborhood is identified. The spatial distribution of the neighboring pixels that have been classified in the first step is now taken into consideration with the class and transitional probabilities. This processing step allows relaxation of the model signature parameters (mean and covariance) to help classification of those pixels that are in the neighborhood of already classified pixels. The relaxation results in additional pixels being classified in this step.

After the relaxation step, there may still be some unclassified pixels that belong to the edges of tissue regions. Pixels that exist on tissue boundaries will have a neighborhood composed of pixels in different classes. Because of limited resolution, the pixel itself may be a mixture of two or more tissue classes. Therefore, for boundary pixels, it is necessary to perform unmixing. Unmixing is the process of determining the set of classes, and the contribution of each, that compose a boundary or a mixed pixel. The unmixing process uses the labels of the already classified neighborhood pixels to determine the candidate set of classes that compose the unclassified pixel. Using the background class model signatures, mixing model analysis is performed for the unclassified pixels. The unmixing analysis can be based on a linear mixing model (22).

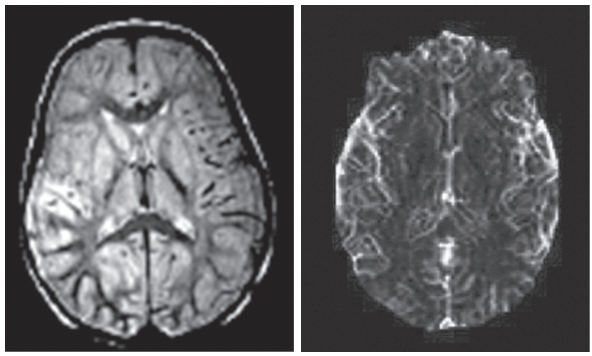
Using the background classes, the linear least squares method is used to determine the fraction of the candidate background classes in an unclassified pixel. This analysis is particularly useful in classifying pixels belonging to the boundary of two or more connected tissue regions. The linear mixing model can be expressed as:

$$\mathbf{R} = \mathbf{e}\boldsymbol{C} \tag{10.46}$$

where $\mathbf{R}$ is multiparameter vector of the measured values of a pixel, $\mathbf{e}$ is the matrix consisting of all signature vectors of candidate background classes, and $\boldsymbol{C}$ is the set of unknown coefficients representing the fraction of the pixel corresponding to each candidate background class. The least squared error solution can be given by

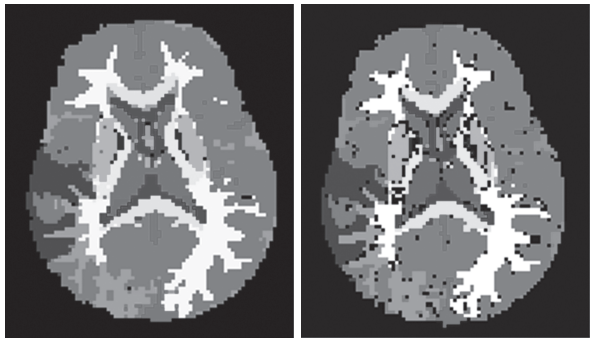
$$\boldsymbol{C} = (\mathbf{e}^{\mathrm{T}}\mathbf{e})^{-1}\mathbf{e}^{\mathrm{T}}\mathbf{R}. \tag{10.47}$$

If all coefficients of $\boldsymbol{C}$ are not positive, the candidate background list is renewed by eliminating the background class with minimum coefficient value. Only coefficients meeting a certain threshold requirement, such as 0.25 (which means that the minimum acceptable fraction for a background class is 25% of the pixel), are kept in the final analysis.



(a) (b)

Figure 10.10 (a) Proton density MR and (b) perfusion image of a patient 48 h after stroke.



(a) (b)

Figure 10.11 Results of MAS method with 4×4 pixel probability cell size and 4 pixel wide averaging. (a) Pixel classification as obtained on the basis of maximum probability and (b) as obtained with $P > 0.9$.

After the mixing analysis is completed, all remaining unlabeled pixels are examined for pathology detection and identification if a database of pathology signatures exists. If such a database does not exist or if an unlabeled pixel defies pathology classification, a new class is created. Figure 10.10 shows the proton density and perfusion MR brain images of a patient 48 hours after a stroke. This image was segmented using a model of 15 tissue classes. The model was developed for 15 classes of normal tissues with no information for stroke-affected lesion and edema tissues. The MAS method developed three new classes and segmented the image into 18 classes. The segmented image, using different classification probability criteria, are shown in Figure 10.11.

10.4.2 Image Segmentation Using Neural Networks

Neural networks provide another pixel classification paradigm that can be used for image segmentation (10, 26, 27). Neural networks do not require underlying class probability distribution for accurate classification. Rather, the decision boundaries

for pixel classification are adapted through an iterative training process. Neural network-based segmentation approaches may provide good results for medical images with considerable variance in structures of interest. For example, angiographic images show a significant variation in arterial structures and are therefore difficult to segment. The variation in image quality among various angiograms and introduction of noise in the course of image acquisition emphasizes the importance of an adaptive, nonparametric segmentation method. Neural network paradigms such as backpropagation, radial basis function (RBF), and self-organizing feature maps have been used to segment medical images (10, 26, 27).

Neural networks learn from examples in the training set in which the pixel classification task has already been performed using manual methods. A nonlinear mapping function between the input features and the desired output for labeled examples is learned by neural networks without using any parameterization (34–38). After the learning process, a pixel in a new image can be classified for segmentation by the neural network.

It is important to select a useful set of features to be provided to the neural network as input data for classification. The selection of training examples is also very important, as they should represent a reasonably complete statistical distribution of the input data. The structure of the network and the distribution of training examples play a major role in determining its performance for accuracy, generalization, and robustness. In its simplest form, the input to a neural network can be the gray values of pixels in a predefined neighborhood in the image. Thus, the network can classify the center pixel of the neighborhood based on the information of the entire set of pixels in the corresponding neighborhood. As the neighborhood window is translated in the image, the pixels in the central locations of the translated neighborhoods are classified.

10.4.2.1 Backpropagation Neural Network for Classification The backpropagation network is the most commonly used neural network in signal processing and classification applications. It uses a set of interconnected neural elements that process the information in a layered manner. A computational neural element, also called perceptron, provides the output as a thresholded weighed sum of all inputs. The basic function of the neural element, as shown in Figure 10.12, is analogous to the synaptic activities of a biological neuron. In a layered network structure, the neural element may receive its input from an input vector or from other neural elements. A weighted sum of these inputs constitutes the argument of a nonlinear activation function such as a sigmoidal function. The resulting thresholded value of the activation function is the output of the neural element. The output is distributed along weighted connections to other neural elements.

To learn a specific pattern of input vectors for classification, an iterative learning algorithm, such as the least mean square (LMS) algorithm, often called the Widrow–Hoff delta rule (35), is used with a set of preclassified training examples that are labeled with the input vectors and their respective class outputs. For example, if there are two output classes for classification of input vectors, the weighted sum of all input vectors may be thresholded to a binary value, 0 or 1. The output 0 represents class 1, while the output 1 represents class 2. The learning algorithm repeatedly presents input vectors of the training set to the network and forces the

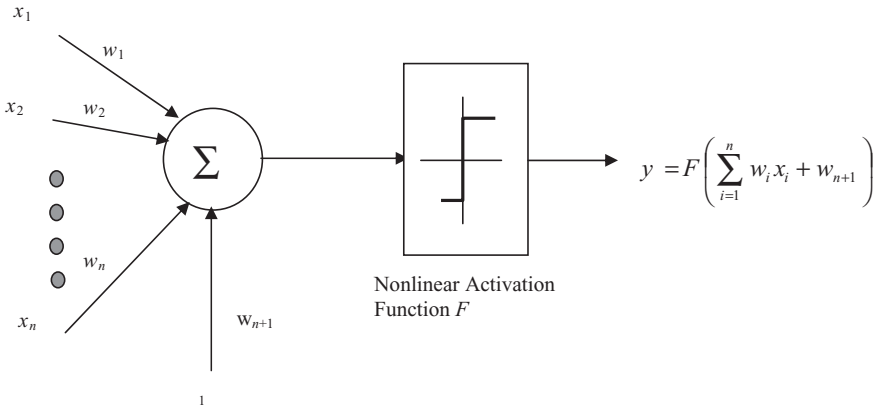


Figure 10.12 A basic computational neural element or perceptron for classification.

network output to produce the respective classification output. Once the network converges on all training examples to produce the respective desired classification outputs, the network is used to sort new input vectors into the learned classes.

The computational output of a neural element can be expressed as

$$y = F\left(\sum_{i=1}^{n} w_i x_i + w_{n+1}\right) \tag{10.48}$$

where F is a nonlinear activation function that is used to threshold the weighted sum of inputs x_i, and w_i is the respective weight. A bias is added to the element as w_{n+1}, as shown in Figure 10.12.

Let us assume a multilayer feed-forward neural network with L layers of N neural elements (perceptrons) in each layer such that

$$\mathbf{y}^{(k)} = F\left(\mathbf{W}^{\mathbf{k}}\mathbf{y}^{(k-1)}\right) \quad \text{for } k = 1, 2, \ldots L \tag{10.49}$$

where $\mathbf{y}^{(k)}$ is the output of the kth layer neural elements with $k = 0$ representing the input layer, and $W^{(k)}$ is the weight matrix for the kth layer such that

$$\mathbf{y}^{(0)} = \begin{bmatrix} x_1 \\ x_2 \\ . \\ x_N \\ 1 \end{bmatrix}; \quad \mathbf{y}^{(k)} = \begin{bmatrix} y_1^{(k)} \\ y_2^{(k)} \\ . \\ y_N^{(k)} \\ y_{(N+1)}^{(\mathbf{k})} \end{bmatrix}$$

and

$$\mathbf{W}^{(k)} = \begin{bmatrix} w_{11}^{(k)} & w_{12}^{(k)} & . & w_{1N}^{(k)} & w_{1(N+1)}^{(k)} \\ w_{21}^{(k)} & w_{22}^{(k)} & . & w_{2N}^{(k)} & w_{2(N+1)}^{k} \\ . & . & . & . & . \\ w_{N1}^{(k)} & w_{N2}^{(k)} & . & w_{NN}^{(k)} & w_{N(N+1)}^{(k)} \\ w_{(N+1)1}^{(k)} & w_{(N+1)2}^{(k)} & . & w_{(N+1)N}^{(k)} & w_{(N+1)(N+1)}^{(k)} \end{bmatrix}. \tag{10.50}$$

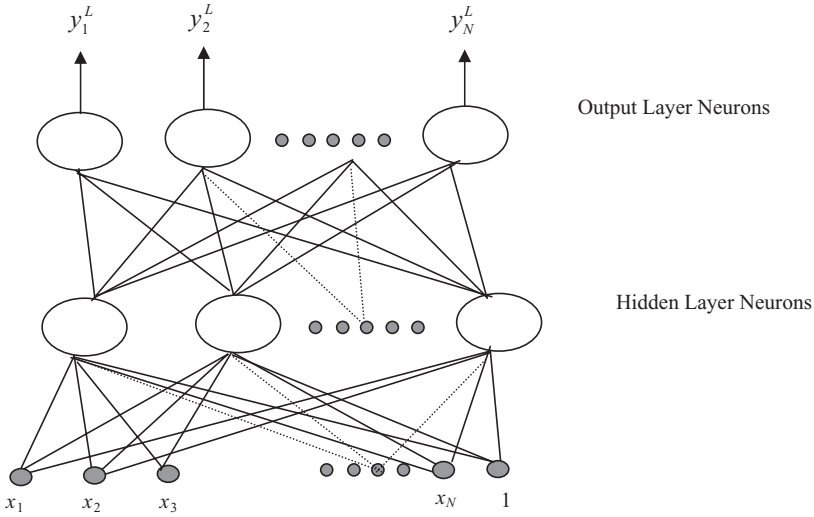


Figure 10.13 A feed-forward backpropagation neural network with one hidden layer.

The neural network is trained by presenting classified examples of input and output patterns. Each example consists of the input and output vectors $\{\mathbf{y}^{(0)}, \mathbf{y}^L\}$ or $\{\mathbf{x}, \mathbf{y}^L\}$ that are encoded for the desired classes. The objective of the training is to determine a weight matrix that would provide the desired output, respectively, for each input vector in the training set. The LMS error algorithm (35, 36) can be implemented to train a feed-forward neural network (Fig. 10.13) using the following steps:

1. Assign random weights in the range of [−1, +1] to all weights w_{ij}^k.
2. For each classified pattern pair $\{\mathbf{y}^{(0)}, \mathbf{y}^L\}$ in the training set, do the following steps:
   a. Compute the output values of each neural element using the current weight matrix.
   b. Find the error $\mathbf{e}^{(k)}$ between the computed output vector and the desired output vector for the classified pattern pair.
   c. Adjust the weight matrix using the change $\Delta\mathbf{W}^{(k)}$ computed as

$$\Delta W^{(k)} = \alpha e^{(k)}[y^{(k-1)}] \text{ for all layers } k = 1, .., L$$

   where α is the learning rate that can set be set between 0 and 1.
3. Repeat step 2 for all classified pattern pairs in the training set until the error vector for each training example is sufficiently low or zero.

The nonlinear activation function is an important consideration in computing the error vector for each classified pattern pair in the training set. A sigmoidal activation function can be used as

$$F(y) = \frac{1}{1 + e^{-y}}. \tag{10.51}$$

The above described gradient descent algorithm for training a feed-forward neural network, also called backpropagation neural network (BPNN), is sensitive to the selection of initial weights and noise in the training set that can cause the algorithm to get stuck in local minima in the solution pace. This causes a poor generalization performance of the network when it is used to classify new patterns. Another problem with the BPNN is to find the optimal network architecture with the consideration of the optimal number of hidden layers and neural elements in each of them. A cascade-correlation neural network architecture has been proposed by Fahlman (36) that finds the best architecture by correlating the error patterns with the inclusion or deletion of neural elements in the hidden layers based on the learning vectors in the training set.

10.4.2.2 The RBF Network The RBF network is based on the principle of regularization theory for function approximation. Unlike the backpropagation network, it does not suffer from problems such as sticking in the local minimum and sensitivity to the network architecture. In addition, RBF networks provide more reliable and reproducible results (37–39).

Since the RBF network is a special class of regularization networks, it provides better generalization and estimation properties for function approximation applications than traditional feed-forward neural architectures (37–39). The basic RBF network contains an input layer for input signal distribution, a hidden layer to provide the RBF processing, and an output layer (Fig. 10.14). The RBF representation can be obtained by clustering the input training data to obtain the centroid vector, $\mathbf{c}_i$ for all clusters. An RBF (generally a Gaussian function) is applied to the Euclidean distance between the input vector, $\mathbf{x}_j$, and its own centroid. The output of each RBF unit is then weighted and summed through a linear combiner to provide the final output of the network. The final output of the network $f(\mathbf{x})$ can be given as:

$$f(\mathbf{x}) = \sum_{i=1}^{K} w_i \Phi(\mathbf{x}, i)$$

$$\text{with } \Phi(\mathbf{x}, \mathrm{i}) = \frac{\exp\left(-\dfrac{\|\mathbf{x}-\mathbf{c}_i\|}{2\sigma_i}\right)}{\displaystyle\sum_{j=1}^{\mathrm{K}} \exp\left(-\frac{\|\mathbf{x}-\mathbf{c}_j\|}{2\sigma_j}\right)} \tag{10.52}$$

where $\mathbf{c}_i$ represents the K centers, σ_i represents the satandard deviation of ith cluster, and $\mathbf{w}_i$ represents the K weighted connections from the hidden layer units to the output unit.

Within the basic topology, the system can be represented in state-space by $\mathbf{y} = (\mathbf{F}\ \mathbf{w})$ where $\mathbf{F}$ is the matrix of activation functions

$$F = \begin{pmatrix} \Phi(\mathbf{x}_1, 1) & \ldots & \Phi(\mathbf{x}_1, \mathrm{K}) \\ \ldots & \ldots & \ldots \\ \Phi(\mathbf{x}_\mathrm{N}, 1) & \ldots & \Phi(\mathbf{x}_\mathrm{N}, \mathrm{K}) \end{pmatrix} \tag{10.53}$$

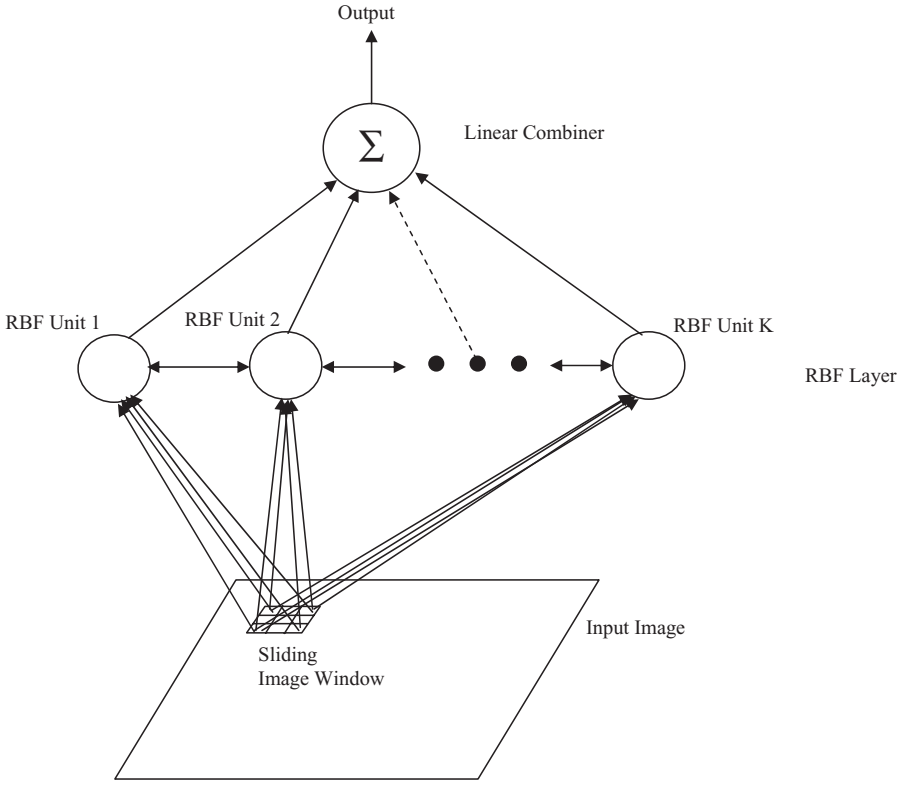


Figure 10.14 An RBF network classifier for image segmentation.

The weight can be calculated from the Moore–Penrose inverse (33)

$$\mathbf{w} = (\mathrm{F}^{\mathrm{T}} \cdot \mathrm{F} + \alpha \cdot I)^{-1} \cdot \mathrm{F}^{\mathrm{T}} \cdot \mathbf{y} \tag{10.54}$$

where α is a small value, such as 0.1.

It has been shown (39) that the matrix inversion would be singular if repetitive data is applied to the inputs of the neural system during training. The major issues for implementation of the RBF network are the location of the centroids and the structure of the RBF. The locations of the RBF depend on the data being presented to the network. The optimal number of clusters can be obtained from the statistical evaluation of the population density and variance parameters of initial clusters through an adaptive *k*-means or fuzzy clustering method (31–33).

10.4.2.3 Segmentation of Arterial Structure in Digital Subtraction Angiograms The arterial structure in a digital subtraction angiogram is characterized by fading ridges and a nonuniform distribution of gray values that makes the segmentation task difficult. Specific features with information about gray values, edges, and local contrast of the pixels are needed for effective segmentation using

a neural network-based classifier (26, 27). The edge features, including ridges, can be obtained using a second-order derivative function such as Laplacian or LOG as described in Chapter 9. The contrast value of a pixel can be simply computed from the largest difference in gray value between a pixel and its 8-connected neighbors. Alternately, feature-adaptive neighborhood processing can be used to obtain a contrast value of the central pixel as used in adaptive feature enhancement method discussed in Chapter 9. In addition, a multiresolution approach can be applied to improve accuracy and efficiency (26).

The feature vector consists of the gray values of the center pixel and its neighbors, combined with the contrast value at the center pixel. The ridge information can be obtained from the first- and second-order derivative operations. A pixel is labeled as a ridge if the first derivative taken along the direction of the second derivative has a zero crossing sufficiently close to the center pixel. The contrast information is computed using the feature adaptive neighborhood processing for the specified neighborhood window. All features are normalized to unit variance to be used in the RBF network classifier.

A training set of example images is prepared with a large number of pixels that are manually classified as belonging to the background or to the arterial structure. Using the respective features of all pixels, feature vectors are prepared and scaled to have a unit variance for fuzzy *c*-means clustering. The centers of the clusters become the locations of RBFs that are used in RBF network-based classification.

Using the fuzzy *c*-means clustering method for the training feature vectors, the RBF network classifier can be developed and trained for classifying pixels into two classes: background and arterial structure. Figure 10.15 shows the results of the RBF network-based arterial segmentation of an angiogram of a pig phantom image.

In the segmentation methods described above, gray values and histogram statistics of pixels in an image have been used as primary features. The histogram statistics may include the mean and variance of the gray values of pixels in the entire image or a specified neighborhood region (40, 41). In region-based segmentation algorithms, additional features may be used to include edge, contrast, and texture information computed from the specified neighborhood of pixels in the image (42). The edge features, including ridges, can be obtained using a second-order derivative function such as Laplacian or LOG as described in Chapter 9. The contrast features are also described in Chapter 9. These features can be effectively used in rule-based systems (16, 23) as well as neural network classifiers for image segmentation (10, 26, 27).

Features representing individual pixels or regions (groups of pixels) in an image play a critical role in image segmentation, analysis, and classification. A general structure of hierarchical knowledge representation for image classification and understanding starts with the properties or features associated with pixels and goes through regions to describe objects present in the image. Features associated with region and object representation for image analysis and understanding are described in the next chapter.

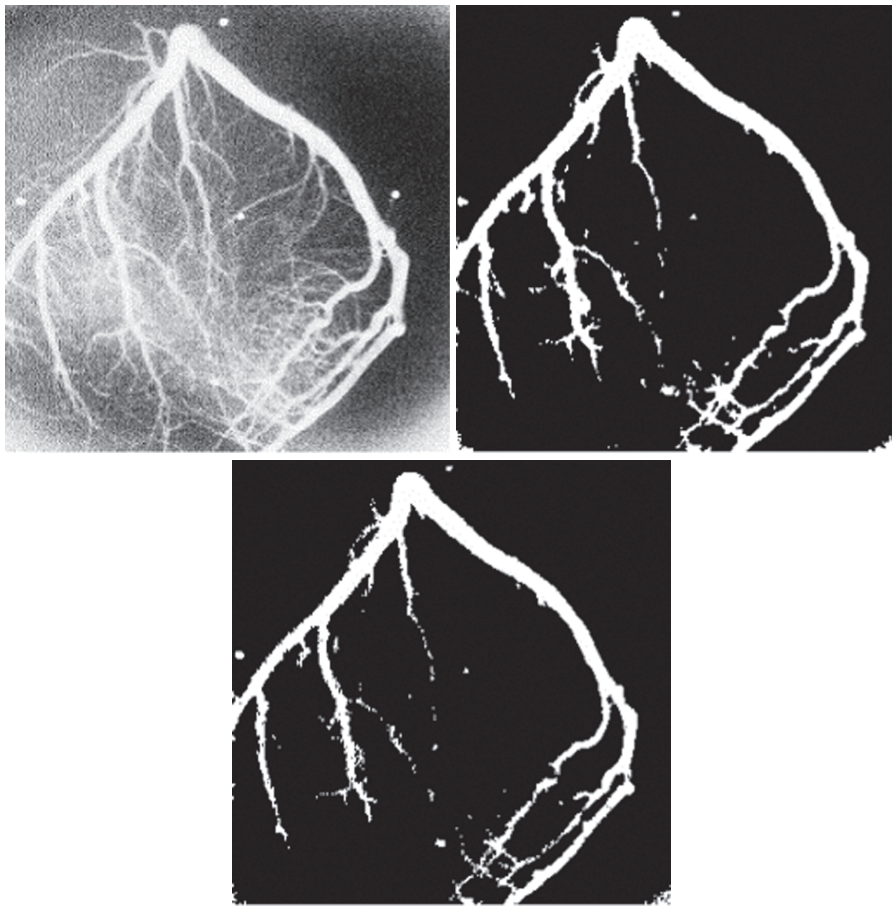


Figure 10.15 RBF segmentation of angiogram data of pig-cast phantom image (top left) using a set of 10 clusters (top right) and 12 clusters (bottom), respectively.

10.5. EXERCISES

10.1. What is the purpose of image segmentation? Why is it a significant step in image analysis?

10.2. What are the major approaches for image segmentation? Describe an image segmentation method of your preference. Explain the reasons for selecting your method.

10.3. How does edge information play a role in region-based segmentation?

10.4. What are the advantages and disadvantages of the Laplacian of Gaussian (LOG) edge detector compared with the Laplacian operator?

10.5. In the MATLAB environment, apply the LOG operator for edge detection on an X-ray mammography image. Threshold the output image at different

levels to extract prominent edges. Comment on the effect of thresholding. How can you select the best threshold for this image?

10.6. Use the same image used in Exercise 10.5 and apply optimal gray value thresholding method for segmentation. Compare the resultant segmented image with the edge-based segmentation obtained in Exercise 10.5.

10.7. Select an MR brain image from the database and apply k-means clustering on the image with $k = 5$. Obtain segmented regions through pixel classification using the clustered classes. Compare the segmented regions with those obtained from the optimal gray value thresholding method.

10.8. Repeat Exercise 10.7 with $k = 9$ and compare your results with those obtained using $k = 5$.

10.9. Using the center of the clusters as obtained in Exercise 10.7, use a region-growing method to obtain segmented regions. Compare the segmented regions with those obtained in Exercise 10.7.

10.10. Write a simple algorithm for image segmentation using a maximum-likelihood-based pixel classification method. Comment on the relative advantages and disadvantages of this approach over the region-growing method for image segmentation.

10.11. Select a classified MR brain image from the database and use it to train a backpropagation neural network (BPNN) for image segmentation through pixel classification. Apply the trained BPNN to classify a new MR brain image for segmentation. Compare the segmented regions with those obtained from gray value thresholding method.

10.12. Explain a fuzzy c-means clustering method. How is it different from the k-means clustering method?

10.13. Select a classified MR brain image from the database and use it to train a radial basis function neural network (RBFNN) for image segmentation through pixel classification. Apply the trained RBFNN to classify a new MR brain image for segmentation. Compare the segmented regions with those obtained from fuzzy c-means clustering ($c = 5$) method.

10.6. REFERENCES

1. R. Jain, R. Kasturi and B.G. Schunck, *Machine Vision*, McGRaw Hill, Inc., New York, 1995.
2. R.C. Gonzalez and R.E. Woods, *Digital Image Processing*, 2nd Edition, Prentice Hall, Englewood Cliffs, NJ, 2002.
3. D. Marr and E.C. Hildreth, "Theory of edge detection," *Proc. R. Soc. Lond. B*, Vol. 207, pp. 187–217, 1980.
4. R.M. Haralick and L.G. Shapiro, "Image segmentation techniques," *Comp. Vis. Graph. Imaging Process.*, Vol. 7, pp. 100–132, 1985.
5. Y. Ohta, *Knowledge Based Interpretation of Output Natural Color Scenes*, Pittman Advanced Pub., New York, 1985.
6. S.A. Stansfield, "ANGY: A rule-based expert system for automatic segmentation of coronary vessels from digital subtracted angiograms," *IEEE Trans. Patten Anal. Mach. Intell.*, Vol. 8, pp. 188–199, 1986.

7. R. Ohlander, K. Price and D.R. Reddy, "Picture segmentation using a recursive region splitting method," *Comp. Vis. Graph. Imaging Process.*, Vol. 8, pp. 313–333, 1978.
8. S. Zucker, "Region growing: Childhood and adolescence," *Comp. Vis. Graph. Imaging Process.*, Vol. 5, pp. 382–399, 1976.
9. A.P. Dhawan and A. Sicsu, "Segmentation of images of skin lesions using color and texture information of surface pigmentation," *Comp. Med. Imaging Graph.*, Vol. 16, pp. 163–177, 1992.
10. A.P. Dhawan and L. Arata, "Segmentation of medical images through competitive learning," *Comp. Methods Prog. Biomed.*, Vol. 40, pp. 203–215, 1993.
11. S.R. Raya, "Low-level segmentation of 3-D magnetic resonance brain images," *IEEE Trans. Med. Imaging*, Vol. 9, pp. 327–337, 1990.
12. Z. Liang, "Tissue classification and segmentation of MR images," *IEEE Eng. Med. Biol. Mag.*, Vol. 12, pp. 81–85, 1993.
13. B.M. Dawant and A.P. Zijdenbos, "Image segmentation," in M. Sonka and J. M. Fitzpatrick (Eds), *Handbook of Medical Imaging, Volume 2: Medical Image Processing and Analysis*, SPIE Press, Bellingham, WA, 2000, pp. 71–128.
14. D.H. Ballard, "Generalizing the Hough transform to detect arbitrary shapes," *Pattern Recognit.*, Vol. 13, pp. 111–122, 1981.
15. M. Bomans, K.H. Hohne, U. Tiede, and M. Riemer, "3-D segmentation of MR images of the head for 3-D display," *IEEE Trans. Med. Imaging*, Vol. 9, pp. 177–183, 1990.
16. S.R. Raya, "Low-level segmentation of 3-D magnetic resonance brain images: A rule based system," *IEEE Trans. Med. Imaging*, Vol. 9, No. 1, pp. 327–337, 1990.
17. H.E. Cline, W.E. Lorensen, R. Kikinis, and F. Jolesz, "Three-dimensional segmentation of MR images of the head using probability and connectivity," *J. Comput. Assist. Tomogr.*, Vol. 14, pp. 1037–1045, 1990.
18. L. Clarke, R. Velthuizen, S. Phuphanich, J. Schellenberg, J. Arrington, and M. Silbiger, "MRI: Stability of three supervised segmentation techniques," *Magn. Reson. Imaging*, Vol. 11, pp. 95–106, 1993.
19. L.O. Hall, A.M. Bensaid, L.P. Clarke, R.P. Velthuizen, M.S. Silbiger, and J.C. Bezdek, "A comparison of neural network and fuzzy clustering techniques in segmenting magnetic resonance images of the brain," *IEEE Trans. Neural Netw.*, Vol. 3, pp. 672–682, 1992.
20. M. Vannier, T. Pilgram, C. Speidel, L. Neumann, D. Rickman, and L. Schertz, "Validation of magnetic resonance imaging (MRI) multispectral tissue classification," *Comput. Med. Imaging Graph.*, Vol. 15, pp. 217–223, 1991.
21. H.S. Choi, D.R. Haynor, and Y. Kim, "Partial volume tissue classification of multichannel magnetic resonance images—A mixed model," *IEEE Trans. Med. Imaging*, Vol. 10, pp. 395–407, 1991.
22. A. Zavaljevski, A.P. Dhawan, S. Holland, W. Ball, M. Giskill-Shipley, J. Johnson, and S. Dunn, "Multispectral MR brain image classification," *Comput. Med. Imaging Graph. Image Process.*, Vol. 24, pp. 87–98, 2000.
23. A.M. Nazif and M.D. Levine, "Low-level image segmentation: An expert system," *IEEE Trans. Pattern Anal. Mach. Intell.*, Vol. 6, pp. 555–577, 1984.
24. L.K. Arata, A.P. Dhawan, A.V. Levy, J. Broderick, and M. Gaskil, Three-dimensional anatomical model based segmentation of MR brain images through prinicpal axes registration," *IEEE Trans. Biomed. Eng.*, Vol. 42, pp. 1069–1078, 1995.
25. L. Xu, M. Jackowski, A. Goshtasby, C. Yu, D. Roseman, A.P. Dhawan, and S. Bines, "Segmentation of skin cancer images," *Image Vis. Comput.*, Vol. 17, pp. 65–74, 1999.
26. A. Sarwal and A.P. Dhawan, "Segmentation of coronary arteriograms through radial basis function neural network," *J. Comput. Inf. Technol.*, Vol. 6, pp. 135–148, 1998.
27. M. Ozkan, B.M. Dawant, and R.J. Maciunas, "Neural-network-based segmentation of multi-modal medical images: A comparative and prospective study," *IEEE Trans. Med. Imaging*, Vol. 12, pp. 534–544, Sept. 1993.
28. N.J. Nilson, *Principles of Artificial Intelligence*, Springer Verlag, New York, 1982.
29. P.H. Winston, *Artificial Intelligence*, 3rd Edition, Addison Wesley, Reading, MA, 1992.
30. R.O. Duda and P.E. Hart, *Pattern Classification and Scene Analysis*, John Wiley & Sons, New York, 1973.

31. T. Kanungo, D.M. Mount, N.S. Netanvahu, C.D. Piatko, R. Silverman, and A.Y. Xu, "An efficient k-means algorithm: Analysis and implementation," *IEEE Trans. Pattern Anal. Mach. Intell.*, Vol. 24, pp. 881–892, 2002.
32. X.L. Xie and G. Beni, "A validity measure for fuzzy clustering," *IEEE Trans. Pattern Anal. Mach. Intell.*, Vol. 13(8), pp. 841–847, August 1991.
33. A. Bezdek, *Pattern Recognition with Fuzzy Objective Function Algorithms*, Plenum, New York, 1981.
34. D.L. Pham and J.L. Prince, "Adaptive fuzzy segmentation of magnetic resonance images," *IEEE Trans. Med. Imaging*, Vol. 18, pp. 737–752, 1999.
35. J.M. Zurada, *Introduction to Artificial Neural Systems*, West Publishing Co., Boston, 1992.
36. S.E. Fahlman and C. Lebeire, *The Cascade-Correlation Learning Architecture*, Tech. Report, School of Computer Science, Carnegie Mellon University, 1990.
37. S. Chen, C.F.N. Cowan, and P.M. Grant, "Orthogonal least squares learning for radial basis function networks," *IEEE Trans. Neural Netw.*, Vol. 2, No. 2, pp. 302–309, 1991.
38. T. Poggio and F. Girosi, "Networks for approximation and learning," *Proc. IEEE*, Vol. 78, No. 9, pp.1481–1497, 1990.
39. I.R.H. Jacobson, "Radial basis functions: A survey and new results," In D. C. Handscomb (Ed), *The Mathematics of Surfaces III*, Clarendon Press, Gloucestershire, 1989, pp. 115–133.
40. S. Loncaric, A.P. Dhawan, T. Brott, and J. Broderick, "3-D image analysis of intracerebral brain hemorrhage," *Comput. Methods Prog. Biomed.*, Vol. 46, pp. 207–216, 1995.
41. J. Broderick, S. Narayan, A.P. Dhawan, M. Gaskil, and J. Khouri, "Ventricular measurement of multifocal brain lesions: Implications for treatment trials of vascular dementia and multiple sclerosis," *Neuroimaging*, Vol. 6, pp. 36–43, 1996.
42. P. Schmid, "Segmentation of digitized dermatoscopic images by two-dimensional color clustering," *IEEE Trans. Med. Imaging*, Vol. 18, pp. 164–171, 1999.