Applications of Quadtree, Octree, and Binary Tree Decomposition Techniques to Shape Analysis and Pattern Recognition

B. B. CHAUDHURI, MEMBER, IEEE

Abstract—**The binary tree, quadtree, and octree decomposition techniques are widely used in computer graphics and image processing problems. Here, the techniques are reexamined for pattern recognition and shape analysis applications. It has been shown that the quadtree and octree techniques can be used to find the shape hull of a set of points in space while their *n*-dimensional generalization can be used for divisive hierarchical clustering. Similarly, an *n*-dimensional binary tree decomposition of feature space can be used for efficient pattern classifier design. Illustrative examples are presented to show the usefulness and efficiency of these hierarchical decomposition techniques.**

Index Terms—**Binary tree, clustering, octree, pattern classification, quadtree.**

I. Introduction

THE quadtree, octree, and binary tree decomposition methods [1]–[5] are widely used in two- and three-dimensional image processing and computer graphics problems. Some of the application areas involve the image data structure [2]–[5], region representation and picture segmentation [6], [7], genus evaluation and component labeling [8], [9], image smoothing and enhancement [10], [11], data compression [12], [13], and medial axis transformation [14]. The techniques are generalized in higher dimensions for applications that involve the search of point data [15], [16]. Applications of these hierarchical techniques in geographic information systems are also reported [17]. In general, the tree structures are based on the simple principle of divide and conquer, and they may be implemented conveniently in software and hardware. It is the purpose of this paper to explore the applications of these powerful techniques to other areas such as pattern recognition and shape analysis. In particular, the present paper deals with the problems of clustering, classifier design, and the shape hull of a point set in space.

Consider a set of S and N points or dots in n-dimensional Euclidean space $\mathbb{R}^n$. In pattern recognition problems, S may define a set of sample patterns from one or more classes and the problem is to cluster them into a specified or unspecified number of clusters. Again, in computer graphics and image processing problems, S may represent a set of pixels in two or three dimensions, and the problem is to find the perceptual shape of S.

Manuscript received November 12, 1984; revised May 15, 1985. Recommended for acceptance by S. L. Tanimoto.

The author is with the Electronics and Communication Sciences Unit, Indian Statistical Institute, Calcutta 700 035, India.

It is convenient to describe the perceptual shape if the border of S in the form of a polygon (for $n = 2$) or polyhedron (for $n = 3$) is known. Apart from carrying important shape information, the border is useful in normalization, set estimation, intrinsic dimensionality detection, and other related problems [18]. The convex hull polygon (polyhedron) containing S is a unique polygon (polyhedron) that may be used for the purpose. However, it does not always represent the perceptual border of a given S. The perceptual border is the border perceived because of the relative locations of the dots in S and for a fairly densely and uniformly distributed pattern, any human observer is quick to see it without much ambiguity. If the point pattern is perceived to have concavities, then the convex hull does not represent the shape hull properly. However, Jervis [19] presents several algorithms for a shape hull which are based either on taking the union of convex hulls of subsets of S or on the shared nearest neighborhood tests. Fairfield [20] proposed a method which starts out with finding the closest point Voronoi diagram and the Delaunay triangulation [21]. A more generalized method is given by Edelsbrunner *et al.* [22]. For $n = 2$, the complexity of the algorithm is essentially $O(N \log N)$. No work is known to be reported for $n = 3$, but the complexity is expected to be greater since finding Voronoi diagrams in three dimensions requires more computation than $O(N \log N)$.

A simple hierarchical method of finding the shape hull of a point set S is given in Section II that is based on the quadtree and octree partitioning of space. In this method, a unique square (or cube) S_o is defined to contain S. By partitioning S_o, smaller squares (or cubes), called cells, are formed. A cell is called nonempty if it contains a member of S. At a given level of partitioning, the border is defined as the border of the connected components of nonempty cells. The method can be executed in $O(N)$ complexity at any finite level of hierarchy for both $n = 2$ and $n = 3$.

A given S in $\mathbb{R}^n$ can be clustered either hierarchically or nonhierarchically into $N_c \leq N$ clusters. If N_c is fixed and known *a priori*, the nonhierarchical methods appear suitable for clustering. For a variable N_c ranging between 1 and N, the hierarchical techniques are commonly used. A hierarchical technique may be divisive or agglomerative depending on whether N_c is nondecreasing with an increase in the level of hierarchy or not [23]. The divisive

techniques are less popular because of their computer complexity.

In Section III, a computationally attractive divisive clustering technique is proposed. It is based on the multidimensional generalization of partitioning methods described in Section II. The basic idea is to treat each connected component of cells as a cluster at the given level of hierarchy. The technique can be executed in $O(N)$ complexity for a finite n.

The tree decomposition technique can also be applied to the problem of pattern classification. In a pattern classification problem, the n-dimensional pattern feature space is partitioned by decision boundaries into decision regions corresponding to the pattern classes. A decision function for each region or a discriminant function for the boundary between each pair of classes is found. The functions are evaluated on the feature vector of an unknown pattern to decide for its class status.

This strategy is convenient and computationally attractive if the discriminant is either a linear or a quadratic function. However, many practical problems demand higher order polynomial discriminants and, correspondingly, the technique becomes very expensive both at the training as well as the recognition phase. The situation becomes more complicated if a class contains more than one component of decision region and a component is topologically multiply connected. A typical example is given in Fig. 9 of Section IV where it may be more convenient to map the decision space in an efficiently coded form so that an unknown pattern can be quickly classified using the codebook. It is shown in Section IV that a generalization of the binary tree decomposition technique is very useful in mapping the feature space. Subsequently, the mapped feature space can be used for computationally efficient pattern recognition.

II. Hierarchical Partitioning and Shape Hull Determination

A. Partitioning Method

The partitioning method starts out with defining the smallest hypercube S_o containing all points of S. Let x_k denote the kth coordinate and $x_k(\max)$ and $x_k(\min)$ denote, respectively, the maximum and minimum of x_k values of the data points along kth coordinate. Then, a hypercube with sides parallel to the coordinate axes and sidelength $L > \mathrm{Sup}_k|x_k(\max) - x_k(\min)|$ may be constructed for S_o. The hypercube S_o may be assumed to be a closed space set.

The hierarchical partitioning process is such that at any level of hierarchy, each hypercube is partitioned into 2^n smaller hypercubes of equal size by hyperplanes perpendicular to the axes. In two and three dimensions, it is equivalent to the quadtree and octree decomposition techniques, respectively.

For a partitioning factor of 2^n, the original hypercube S_o of side L is decomposed into 2^{ni} equal hypercubes (or ith level cells) so that at any $i > o$ level, an n-dimensional array of 2^{ni} cells are formed. Let an element of the array be denoted by $(a_1, a_2, \cdots, a_k, \cdots, a_n)$ where a_k may range from 1 to 2^i for any k.

Definition 2.1: If a cell contains at least one point of S, then it is a *nonempty* cell. Otherwise, a cell is *empty.*

If a point falls on the partitioning hyperplane or on the intersection of two or more partitioning hyperplanes, then the tie is broken arbitrarily. Also, it is assumed that the cells outside S_o are all empty at any level of hierarchy.

It is clear that any empty cell of ith level is partitioned into empty cells at all subsequent levels, but a nonempty cell may be partitioned into nonempty cells or a mixture of empty and nonempty cells.

Definition 2.2: Any two cells are p_1 *neighbors* if they meet at a point, a line, a plane, or a hyperplane of k dimensions where $3 \leq k \leq n - 1$. Any two cells are p_2 *neighbors* if they meet only at a hyperplane of $n - 2$ dimensions. Two nonempty cells are q *neighbors* if they are p_1 neighbors. Two empty cells are q *neighbors* if they are p_2 neighbors.

The p_1 and p_2 neighborhoods are similar to the 8 and 4 neighborhoods, respectively, in a two-tone image [24].

Definition 2.3: Two nonempty (empty) cells c_1 and c_r are connected by a *path* of nonempty (empty) cells given by a sequence, say, $c_1, c_2, \cdots, c_j \cdots, c_r$ so that c_{j-1} is a q neighbor of c_j for any $2 \leq j \leq r$.

Definition 2.4: A *maximally connected nonempty (empty) region* is a subset of nonempty (empty) cells so that each cell is connected by a path to all of the rest of the cells of the subset only.

The process of hierarchical decomposition is illustrated in two dimensions for $i = 2$ in Fig. 1 where the cells are numbered as 1, 2, $\cdots$, 16. Here the cells 1, 5, 6, 9, and 13 are empty. The rest are nonempty cells. There is only one maximally connected region at $i = 2$ consisting of all the nonempty cells.

For any type of data set S, there exists only one region at the lowest levels $i = 0, 1$ of the hierarchy. Usually, the number of regions grows with i. For example, cell 2 of Fig. 1 is partitioned into 16 cells at $i = 4$, leading to three nonempty regions each consisting of one nonempty cell. In fact, given any data set, the number of regions is N for a sufficiently large i.

B. Shape Hull Determination

The border of nonempty regions and their holes constitutes the shape outline of S at the given level of hierarchy. Informally speaking, a hole is an empty region surrounded by a nonempty region. The border of the regions and their holes can be found by identifying the border cells defined below.

Definition 2.5: A nonempty cell is a *border cell* if it is the p_2 neighbor of at least one empty cell. A nonempty cell is an *interior cell* if it is not a border cell.

It is understood that the shape hull is meaningful mainly in two and three dimensions. A border cell contains border sides and border surfaces in two and three dimensions, respectively. Let the cells be closed subsets in space.

Definition 2.6: A *border side* is the intersection of a

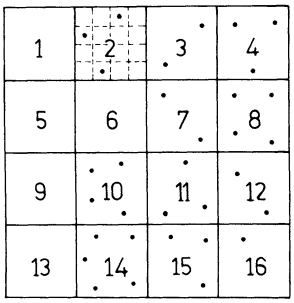


Fig. 1. The quadrant partitioning technique.

border cell with one of its neighboring empty cells in two dimensions. Similarly, a *border surface* is the intersection of a border cell with one of its neighboring empty cells in three dimensions. The union of all the border sides (surfaces) of a maximally connected region is called its *border*.

It is clear that a border cell may have at most four border sides in two dimensions and six border surfaces in three dimensions.

Since the nonempty cells are assumed to be closed subsets in space, each maximally connected nonempty region is also a closed subset in space. The union of all such nonempty regions that contains all the points of S may be termed as the shape hull of S. The shape hull may be described in terms of the borders of the nonempty regions.

The maximally connected nonempty regions, together with their border, specify the shape hull of the point set. Apart from the determination of the shape hull, the present method can be used to describe the shape topologically. In fact, the holes and nonempty connected regions can be conveniently counted and the Euler number may be derived [8], [9]. Also, the borders of a nonempty region may be distinguished as exterior and interior borders (borders of the holes).

C. Computer Algorithms and Complexity

For simplicity and convenience, we consider $n = 2$ in describing our algorithms.

Partitioning at any level i from $i - 1$ can be conveniently done by defining a two-dimensional array, say, A_i, for the ith level. Let a cell of A_i be designated at (j, k) where j denotes the row and k denotes the column of the array. The partitioning algorithm contains the following steps.

Step 1): Redesignate each cell $(j, k)_{i-1}$ of level $i - 1$ as four cells $(2j - 1, 2k)_i$, $(2j - 1, 2k - 1)_i$, $(2j, 2k - 1)_i$ and $(2j, 2k)_i$ at level i.

Step 2): If the cell $(j, k)_{i-1}$ is nonempty, test which of these four cells contain at least one point of S. Redesignate them as nonempty and the rest as empty.

Step 3): Increase i by 1 and go to step 1.

There are many border following algorithms available in the literature [5]. Most algorithms find the set of border cells rather than a string of border sides and surfaces. An interesting method of finding a boundary from the quadtree representation is given in [6]. In the following, an algorithm is described where the quadtree structure is not specifically utilized. Also, the border is found in the form of a string of border sides so that the crack-coded representation of the border is possible. For brevity, let the border cell and interior cell be called as the B cell and I cell, respectively.

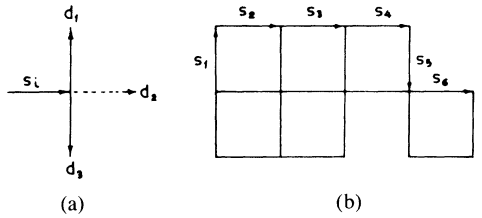


Fig. 2. The method of border following (a) three neighboring directions of a side. (b) Part of the followed border is $S_1, S_2, \cdots, S_6$.

1) Test the neighborhood in A_i and label the B cells and their border sides using definition 2.5 and 2.6, respectively.

2) Specify the first B cell encountered in horizontal scanning and take its border side as the initial side s_1 of a border.

3) For each s_j, see which of its three neighboring directions contain border sides. Choose s_{j+1} by the "rule of border side following" given below unless s_1 is encountered again. Let s_m be the side encountered just before s_1. The sequence $s_1, s_2, \cdots, s_m$ denotes a border.

4) Delete $s_1, s_2, \cdots, s_m$ from the corresponding B cells and convert the B cells having no border side into I cells. Choose any B cells left and take any of its border side, as s_1 for a new border. Go to step 3.

The "rule of border side following" is illustrated through Fig. 2. A border side s_1 may be considered as a vector which has three neighboring directions d_1, d_2, d_3 for border following. If there exists a border side along each of the directions d_1, d_2, and d_3, which makes 90° with s_j and does not belong to the B cell to which s_j belongs, it is taken for s_{j+1}.

In three dimensions, partitioning can be done by defining a three-dimensional array and redesignating each cell of the $i - 1$th level as eight cells at the i level. The cells are labeled as empty and nonempty in a similar manner as above. The border surface of each cell is a square, and each side of this square may be connected to another border surface in one of the seven orientations rather than three of Fig. 2. Instead of sequentially following the border sides as in a two-dimensional case, it is convenient to grow the border surface in all directions, in which case, a stack algorithm may be employed.

Let us now ascertain the computational effort necessary for the procedure discussed above. The change in position coordinate of the points requires an $O(N)$ complexity. Each level of the partitioning process can also be completed in $O(N)$ time. This is because $2^n - 1$ comparisons are necessary to decide which cell contains a point at the current partition, and one point is tested only once. Finally, the determination of border involves the nonempty cells and each nonempty cell is accounted for a finite number of times. At any arbitrary level i, the number of nonempty cells is less than or equal to N. Hence, border finding es-

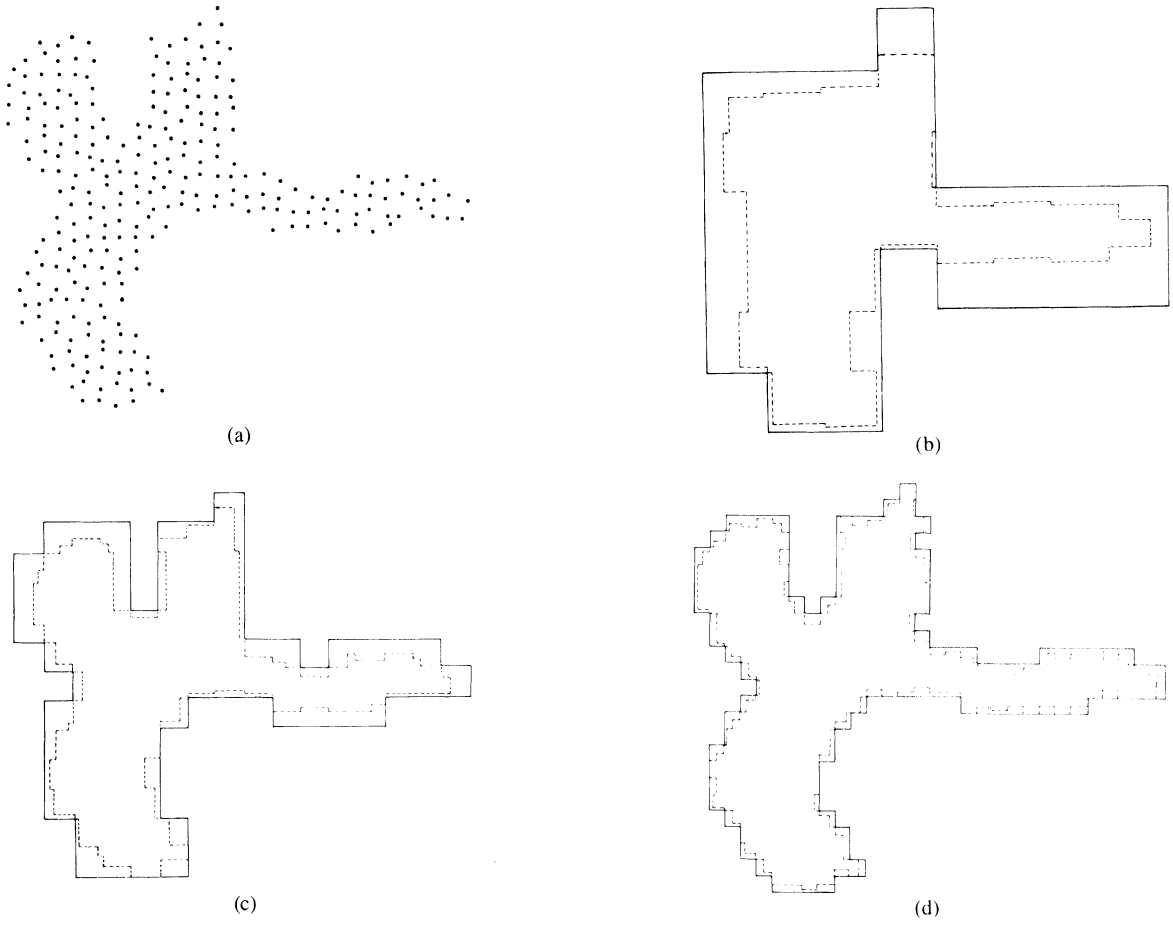


Fig. 3. The hierarchical shape of a point pattern. ———— unrefined border, - - - - - - refined border. (a) The point patten, (b) for $i = 3$, (c) for $i = 4$, and (d) for $i = 5$.

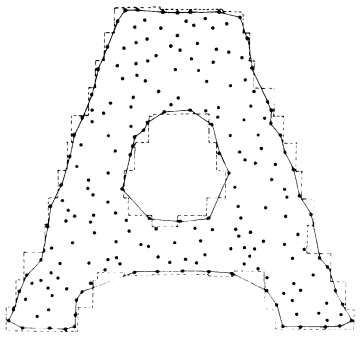


Fig. 4. Border of a point pattern. ——— Edlesbrunner *et al.* [7]. - - - - the method in this paper.

sentially involves $O(N)$ complexity. For a finite number of levels, therefore, the above technique can be run in $O(N)$ time.

D. Experimental Results and Discussion

In order to test the efficiency of the present technique, some perceptually meaningful dot patterns were taken. Two of them are shown in Figs. 3(a) and 4, respectively. The first dot pattern outlines the shape of a chromosome while the second pattern denotes the letter "A" in the English alphabet.

In order to get a better approximation of the shape of the dot pattern, a border refinement technique is utilized. To refine the border at a given level $i = i_0$, we distinguish four types of border (B) cells. We call those having only one border side type 1 B cells, those having two border side type 2 B cells, and so on.

For type 1 cells, the refinement is done by translating the border side toward the cell, so long as all the points belonging to the cell remain inside it. For a type 2 cell, both border sides are translated as in type 1 cells if the sides are adjacent or if there are more than one point inside the cell having their x and y coordinates distinct. Otherwise, only one side is translated. The side whose translation maximally reduces the area of the cell is chosen for the purpose. For a type 3 cell, again, all the border sides are translated if the cell contains more than one point having distinct x and y coordinates. Otherwise, only one side is translated. The choice of side, again, is guided by the maximal reduction of cell area. The situations are illustrated in Fig. 5(a)-(c).

The case of type 4 cells is a little complicated. At first, the corners of the cell which are common to the corners of the other cells are labeled as connecting corners. The rest of the corners are called free corners. There may be one, two, three, or four connecting corners in a type 4 cell. For a cell with four connecting corners, no refinement is done. For three connecting corners, the refinement is done at the free corner by curving out a rectangle

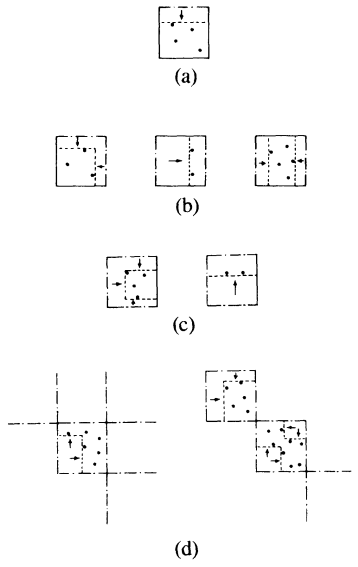


Fig. 5. The method of border refinement. - - • - - border side before refinement. - - - - border side after refinement.

of maximum area as in Fig. 5(d_1). For two connecting corners, the two free corners are chosen. For a cell with one connecting corner, the diagonally opposite free corner is chosen to reduce the area of the cell by curving our rectangle of maximum area as shown in Fig. 5(d_2). The results of using the present technique in Fig. 3(a) are shown in Fig. 3(b), (c), and (d) for different levels of hierarchy. The number of points in this pattern is less than 300, and we propose $i = 5$ as the terminal level of maximum resolution. In fact, at $i = 6$ a large number of small holes appears in the figure and, at a higher level, the region starts breaking up into smaller regions.

It is clear from the figures that a quadtree structure of shape can be created by the empty and nonempty cells at each level of hierarchy. The quadtree structure can be used conveniently in shape matching problems. Let S_o be normalized to have unit area for any S. Then two sets of point patterns may be said to match at ith level if their borders match at that level.

Quite often, it is convenient to use a two-dimensional array representation in digital processor. The nonempty and empty cells may be conceived as black and white pels and many digital two-tone image processing algorithms can be used here with minor modifications. A gray tone appearance can also be given to the nonempty cells if the darkness of a cell is assumed to be proportional to the number of points in the cell.

The present method is computationally cheaper than the method due to Edelsbrunner *et al.* [8]. In their method, however, the hierarchy is continuous on a real-valued parameter α rather than on a discrete-valued i here. Also, the borders in their method are formed by joining some border points found from the Delaunay triangulation. On the other hand, the border in our method has a rectangular crack shape. The point pattern "A" and its border obtained by the two methods are compared in Fig. 4.

III. Application to Clustering

A. Clustering Properties

The above technique may be viewed as that of a divisive hierarchical clustering if we accept the following definition.

Definition 3.1: The points belonging to each maximally connected nonempty region at a given level form a *cluster* of that level.

The divisive technique may be modified to the case when the number N_c of clusters to be obtained is fixed *a priori*, as in the nonhierarchical techniques. The idea is to go up to the level when the number of clusters obtained is greater than or equal to N_c. If it is greater, most similar pairs may be merged unless the number becomes N_c. We are interested here in hierarchical techniques only.

Let us investigate the properties of this clustering method. At any level i, the size of a cell is l_i^n where $l_i = L/2^i$. The diagonal of the cell is $l_{d,i} = (nl_i^2)^{1/2}$. Let C_{hi} and C_{ki} be any two of the clusters formed at level i, and let $d(x, y)$ denote the Euclidean distance between any two data points x and y. Let $D(C_{hi}, C_{ki}) = \text{Inf}[d(x, y), x \in C_{ki}, y \in C_{ki}]$ be the distance between the clusters C_{hi} and C_{ki}. Also, let S_a and S_b be any two sets of data points in the n-dimensional space.

Property 3.1: At any level i, we have $D(C_{hi}, C_{ki}) > l_i$.

Property 3.2: If $S = S_a \cup S_b$ and $D(S_a, S_b) > 2l_{d,i}$, then no cluster is formed at level i that contains points from both S_a and S_b.

The property 3.2 tells that if the distance between two sets of points is greater than $2l_{d,i}$, then they are clustered in separate clusters at level i. The number of separate clusters may be more than two.

According to the definition 3.1, a cluster is formed because of connectivity of the nonempty cells. Instead of using the cell connectivity defined in Section II-A, let us examine the behavior of the clustering in terms of r connectivity of the data where r is a real number [25].

Definition 3.2: A point x is *r connected* to the point y in X if there exists a sequence of points $Z_0, Z_1, Z_2, \cdots, Z_k, Z_{k+1}, Z_0, = x, Z_{k+1} = y, Z_0, Z_1, \cdots, Z_{k+1} \in X$ such that $d(Z_j, Z_{j+1}) \leq r$ for all j. The points of a set X are r connected if all $x, y \in X$ are r connected and there exists at least one point in X whose nearest neighbor in X is r distance away.

Property 3.3: The points of any cluster C_{hi} are, at most, $2l_{d,i}$ connected.

Property 3.4: Any set of data points $S = S_a$ will form a single cluster at level i if all $x \in S_a$ are l_i' connected where $l_i' = l_i - e_i$; $0 < e_i < l_i$.

Property 3.5: A cluster obtained at level i will not be partitioned into more clusters at level $i + 1$ if it is $(l_i - e_i)/2$ connected while it will be partitioned into more clusters if it is r connected where $2l_{d,i+1} < r < 2l_{d,i}$ and $l_{d,i+1} = (nl_i^2/4)^{1/2}$.

For partitioning at level $i + 1$ from level i, the range of connectivity between $(l_i - e_i)/2$ and $2l_{d,i+1}$ is unpredictable.

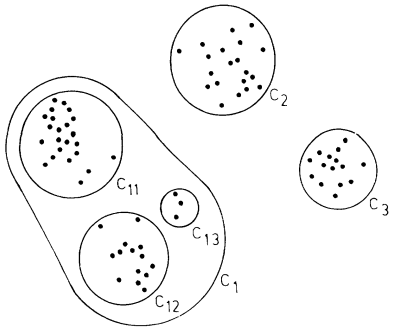


Fig. 6. Clustering of Ruspini's data [26].

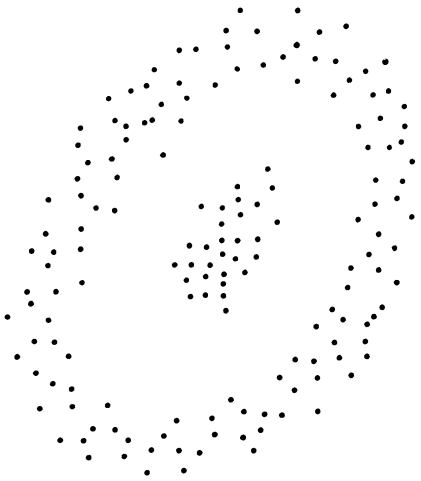


Fig. 7. A cluster enclosed by a cluster.

It is clear from the algorithms in Section II-C that the clustering can be obtained in $O(N)$ time at any level of hierarchy. Practically, the number of meaningful clusters is much less than the number of data points. Also, there should be a smooth transition in the number of clusters with increasing i. In the present technique, this is true up to a certain value of i. The process should be stopped at a level where the number of clusters grows abruptly.

B. Experimental Results and Discussion

The procedure has been tested successfully on different multidimensional data sets. As an example, Ruspini's [26] artificial data set is illustrated in Fig. 6. The data are partitioned into three clusters C_1, C_2, and C_3 at level $i = 3$. At level $i = 4$, the cluster C_1 is partitioned further into three clusters C_{11}, C_{12}, and C_{13}. For $i = 5$, the number of clusters starts growing abruptly and, at $i = 7$, the number of clusters becomes 75 which is equal to the number of data points.

Other typical data, such as the visually perceptible cluster inside a cluster as shown in Fig. 7, can also be discriminated at a level if the distance between the clusters at that level exceeds $2l_{d,i}$. Thus, the two clusters shown in Fig. 7 are clearly separated at $i = 5$. However, it is possible to generate data so that the number of clusters at any level is either greater than or less than two, although there are perceptually two clusters in the data set. There are two possible reasons for such a situation. The primary reason

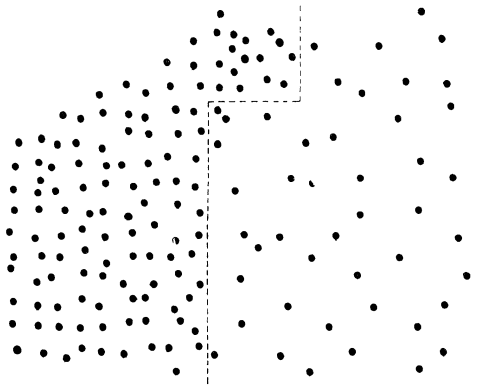


Fig. 8. Two touching clusters of variable density.

is that the cell size reduction, with the increase in level, is logarithmic rather than linear, and the average density of data points may fall in between the cell density of data of any two consecutive levels. Hence, the hierarchical partitioning may not follow up the data clustering structure properly. A secondary reason is that a hypercube is not rotationally symmetric around the center and the diagonal length is much larger than the side length. It is possible to modify the procedure to eliminate both the drawbacks to a limited extent. To get a slow rate of cell size reduction at level i, one may go back to level $i - 2$ and partition each cell into 3^n equal cells by three hyperplanes perpendicular to each of the n axes. If this is called the modified ith level i_{mod}, then the ratio of the cell size in the two cases will be $(\frac{3}{4})^n$. Instead of $i - 2$, one may go down to $i - 3$ and define i_{mod} by partitioning each cell into 3^n, 5^n, or 7^n cells. In those cases, the slow down rates of $(\frac{3}{8})^n$, $(\frac{5}{8})^n$, and $(\frac{7}{8})^n$ are possible. However, the program complexity increases by this modification. The difficulty of the nonsymmetric nature of cell size is eliminated to some extent if the axes are rotated along the diagonal in each plane and results of clusterings are observed. The best result may be hypothesized as the one where the minimum number of clusters is obtained at a given level. We found this hypothesis to be helpful for some data sets.

Another structure that may not be disclosed directly by the present technique is the touching clusters with variable density as shown in Fig. 8. When definition 3.1 is directly applied to the data of Fig. 8, one gets a single cluster up to level 3, while at level 4, the number of clusters becomes quite large. To overcome the problem, we take the cell density histogram and use a threshold to distinguish two types of nonempty cells of different density. The density of the cell is defined as the number of points inside it. Now, instead of "maximally connected region of nonempty cell," the "maximally connected region of nonempty cells of a given density" is defined as a cluster. At level 2 of Fig. 8, the cell density histogram produces two sharp peaks around the densities 4 and 12. The two types of nonempty cells are distinguished by a threshold at density equal to 8 and the two touching clusters are found at level 2.

It is clear that the above modification is equivalent to

viewing the problem as that of finding maximally connected regions in a gray-level image while the original technique is its bilevel version.

IV. Binary Tree Decomposition in Pattern Classification

A. Partitioning and Mapping Technique

Consider the recognition problem of M classes in an n-dimensional feature space. Any point in the space may be represented in terms of the coordinates labeled x_1, x_2, $\cdots$, x_n. Let the decision region of the mth class C_m contain m_r components represented by $S_{m,i}$; $i = 1, m_r$. Consider $S_{m,i}$ as closed subsets in $\mathbb{R}^n$. Let $V_n = \cup_m^M \cup_i^{m_r} S_{m,i}$ be the domain of the feature space in which all the patterns must lie. In general, V^n may take any shape. We assume that V^n is topologically simply connected and its boundary consists of hyperplanes perpendicular to the coordinate axes only. The boundary between the ith component of C_m and the jth component of C_k is given by $Q_{m,k,i,j} = S_{m,i} \cap S_{k,j}$. Then, the decision boundary of the region of C_m may be expressed as

$$Q_m = \bigcup_k^M \bigcup_i^{m_r} \bigcup_j^{k_r} Q_{m,k,i,j}. \tag{1}$$

The above definition permits the decision boundary to have finite thickness if necessary. In a conventional pattern recognizer, the thickness of the decision boundary is assumed to be zero. As described later in this section, the boundary having finite thickness is a more realistic assumption in certain problems.

Let V^n be partitioned into equal halves by a hyperplane perpendicular to x_1. Any of the halves lying entirely within the decision region of any class is left untouched. The mapping of this decision region is done by associating the class in which it lies. On the other hand, if the decision boundary lies within any half, it is partitioned again into equal halves by a hyperplane perpendicular to x_2. Mapping of the halves is done if possible. Otherwise, they are partitioned to halves by a hyperplane perpendicular to x_3. The process may be repeated for t stages so that at the ith stage, the partitioning is done perpendicular to the coordinate x_k where $k = i$ modulo n.

As an example, the class C_1 in Fig. 9 is partitioned for $t = 12$ using the above technique. The largest square in C_1 of Fig. 9 is generated at the 4th level of partitioning. This square is not partitioned further since it lies entirely within C_1. This is true for all the squares except some of the smallest ones which are not partitioned further since they are produced at the terminal level of partitioning.

Consider again the largest square in C_1. Any pattern falling in this square can be classified at the 4th level. It requires four comparisons to know if the coordinates of a pattern lie within this square or not. Once this is done, a tree look down will clarify its class status.

In Fig. 10, the tree structure of the partitioned space of C_1 of Fig. 9 is shown up to the 8th level. In two dimensions, a rectangular space may be partitioned into left and right or upper and lower halves which are represented in Fig. 10 as L, R, U, and D, respectively. When one of these symbols stands alone at a node of the tree, it is not a terminal or leaf node. A leaf node is a decision node if C_1 or $\overline{C}_1$ is associated. Here, C_1 signifies that the corresponding partitioned region is entirely within the decision region of class C_1 while $\overline{C}_1$ signifies that it is entirely outside the decision region of C_1. The decision node corresponding to the largest square in C_1 of Fig. 9 is marked by a circle in Fig. 10.

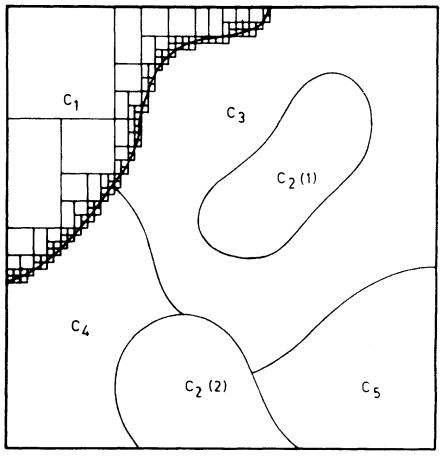


Fig. 9. Decision regions of five pattern classes in two-dimensional feature space.

Before discussing C_1', let us see how the tree look down can be understood as an efficient decision technique. Let each decision node be represented by a unique binary code. The lower the node is in the tree structure, the larger is the code length. Thus, the node marked by a circle in Fig. 10 can be represented by a 4-bit codeword, while the terminal level of the tree may be represented by an 8-bit codeword. Since a larger partitioned rectangle of a decision region is represented by a smaller codeword, it is an efficient feature space coding technique. Now, for uniformly distributed data it leads to a computationally efficient pattern classification. It should be noted that the decision node with a symbol such as $\overline{C}_1$ does not occur in a complete tree structure.

In the actual method of classification, the binary tree structure is stored in the coded form along with the class codes to which the decision nodes belong. Now, the binary decomposition method is used on the position coordinates of a pattern to be classified so that a binary code is generated corresponding to the square in which the patten belongs. Comparing this pattern code bit by bit to the stored coded tree structure, the class status of the pattern is automatically found.

C_1' signifies that the corresponding partitioned rectangle lies partly within and partly outside the decision region of C_1. In other words, the decision boundary of C_1 and some other class falls within this rectangle. It is to be noted that C_1' appears at the terminal level i_t beyond which no space partitioning is allowed. Actually, the smallest squares of Fig. 9 obtained at the 12th level and containing parts of the border of C_1 are candidates for C_1'. Fig. 10 depicts the

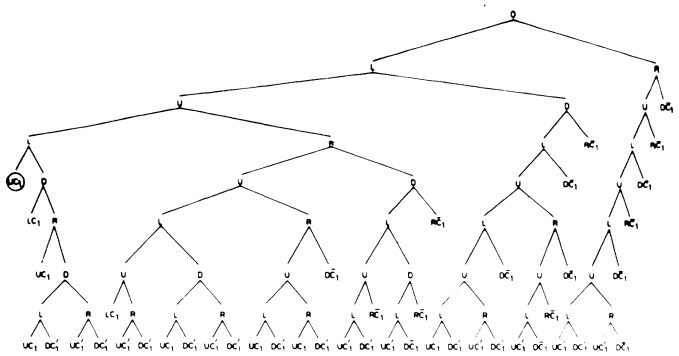
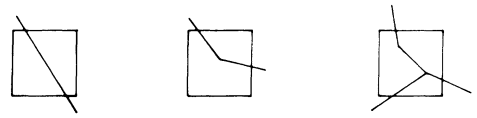


Fig. 10. A binary tree for class C_1 of Fig. 1.



(a) (b) (c)

Fig. 11. A square containing decision regions of two or more pattern classes. (a) Linear boundary between two classes. (b) Piecewise linear boundary between two classes. (c) Piecewise linear boundary between three classes.

C_1's which are produced if the space partitioning is terminated at the 8th level. Some heuristic must be used to decide for the pattern falling in C_1 so that the tree decision codebook is completed.

One simple heuristic may be an arbitrary tie-breaking rule so that any rectangle labeled C_1' that contains a boundary of q classes is attributed to any of these q classes. An alternative may be to classify the patterns among q classes with equal frequency. Yet another method is to attribute a pattern to the class whose decision region within the square is largest. It becomes easier to compute the decision region if the boundary is linear as in Fig. 11.

The presence of rectangles labeled C_1' may be viewed as the decision boundary of C_1 having finite thickness as if formed due to overlap of the decision regions of other classes with that of C_1. Such a boundary broadening is not an impractical case in many situations. For example, if the number of samples at the training phase is not sufficiently large, then the decision parameters may not be obtained with sufficient confidence. It may be more adequate to express the parameters lying within a range, rather than having a single value. Then the decision regions of the classes will overlap to broaden the decision boundary. A similar situation may occur due to imprecision and noise in the feature measurement system. In fact, any measuring system has finite resolution and the telemetric data are usually prone to noise and distortion. Also, in some applications, the pattern classification is conducted on data taken over a long period and the characteristics of the pattern can change slowly with time. One may cope with the situation by broadening the decision boundary of the pattern recognizer. If the boundary is very thick, a fuzzy set theoretic recognition model appears more appropriate than others. The topic will be treated elsewhere in detail.

Whatever might be the recognition heuristic, the effect of decision boundary broadening is to increase the misclassification rate. Let n be a factor of the number of terminal levels i_t so that the broadened boundary is formed by hypercubes, each having diagonal length d_t. Let s_m denote the decision hypersurface for the mth class $1 \leq m \leq M$. Then the hypervolume covered due to decision boundary broadening has the upperbound

$$v \leq d_t \sum_{m=1}^{m} \int_{s_m} ds_m. \tag{2}$$

If the feature domain V^n occupies a hypervolume v_o, then v/v_o is the fraction of space over which a heuristic decision is to be made. If the patterns are evenly distributed, then v/v_o is the fraction of patterns on which additional misclassification is encountered. It is possible to reduce the additional misclassification arbitrarily by increasing i_t and hence reducing v. But an increase in i_t leads to a large tree structure.

From Fig. 9 it is clear that rectangles of different sizes are formed at different levels of partitioning. They may be called as cells in n dimensions. Let n_i be the number of i-level cells formed by the partitioning process and let the hypervolume of an i-level cell be v_i. If δ_{ij} is the average density of the data to be classified to the jth of the i-level cells, then $v_i \sum_{j=1}^{n_i} \delta_{ij}$ is the number of data falling in the i-level cells. The total number of data to be classified is $N = \sum_{i=1}^{t} v_i \sum_{j=1}^{n_i} \delta_{ij}$. Then, the expected number of levels of partitioning necessary to classify a datum is

$$i_{av} = \underset{N \to \infty}{Lt} \frac{\sum_{i=1}^{t} i v_i \sum_{j=1}^{n_i} \delta_{ij}}{N}. \tag{3}$$

The average number of comparisons necessary to ensure class status of a datum is $2i_{av}$. Since comparison of numbers is only involved, it is computationally very attractive.

There is an interesting similarity between the present technique and Wald's sequential decision technique [27]. In the Wald's techniques, the pattern features are tested sequentially until a decision is made. This test is equivalent to observing the pattern sequentially with respect to the coordinates $x_1, x_2, \cdots, x_n$. Similarly, in the present case, the space is partitioned with respect to the coordinates $x_1, x_2, \cdots, x_n$ until a decision about the class containment can be made. It is, therefore, expected that the

cost effectiveness of Wald's technique will be reflected in the present technique.

B. Results and Discussion

To understand the computational advantage of the present method, let us take a simple example of nearest neighbor (*nn*) classifier for five classes with 20 prototypes for each class in two dimensions. For the 1-*nn* classifier, it requires 100 distance computations and 100 comparisons to classify a pattern by this method. Simulated study has shown that a terminal level of $i_t = 25$ is good enough for a feature space with quite complicated (i.e., zig-zag) decision boundary and the decision region of each class having at least two components. In that case, only 50 comparisons are enough even if the decision for each pattern is made at the terminal level. However, the storage requirement is always higher for the present technique.

We considered Fig. 9 as one of the typical cases of simulated two-dimensional feature space with a decision boundary of five classes. The whole space was partitioned and labeled as in class C_1 shown in the figure and the resultant tree structure was stored in a coded form. The terminal level was chosen as $t = 12$. This resulted in the decision boundary broadening that covered 6.37 percent of the decision space. Two-dimensional random numbers with uniform density over the space were used as patterns to be classified. The boundary broadening effect made a 0.35 percent degradation in recognition score for arbitrary tie-breaking heuristic and 0.22 percent for maximum area heuristic. The average number of levels required to recognize an unknown pattern was 7.356. It required 710 locations to store the tree code book.

As a practical example, the recognition of vowels from human speech sound was considered. The vowel data were collected from about 350 commonly used multisyllabic Telugu (a major Indian language) words spoken by five adult male informants. The recordings were made inside an empty auditorium with an AKAI 1710 recorder on TDK tapes and the spectrographic analysis was done on a KAY sonagraph model 7029-A. The first two formant frequencies F_1 and F_2 were chosen as two features. In feature space, the decision boundary of the five vowels a, e, i, o, and u is shown in Fig. 12. The boundary was obtained by equating the potential functions of the neighboring classes. To reduce excessive variation, the actual boundary was smoothed out to take the present form. Using the present decision space mapping technique, the recognition score was 79.3 percent. The reported recognition score of the same data on other techniques lies around 80 percent [28]. The average number of levels required to recognize a pattern was 6.453. The corresponding computational effort is less than that in the Bayes' rule, even if the features are assumed to be multivariate and normally distributed over each of the vowel classes.

In general, the technique is quite useful if the decision boundary of classes does not consist of simple hyperplanes and if there are more than one decision regions for the classes. The binary tree is found to be a good database

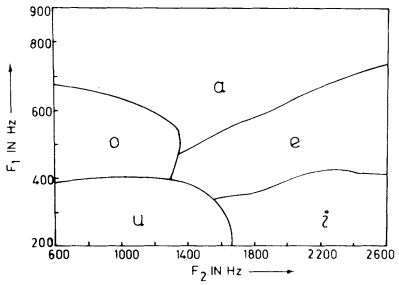


Fig. 12. Decision regions of five Telugu vowels in F_1-F_2 feature plane.

structure for two-tone images. This property can be used advantageously in modifying the decision space, whenever necessary, in a pattern recognizer.

V. Conclusion

The purpose of this paper is to examine the generalization of quadtree, octree, and binary tree decomposition techniques to problems other than image processing and computer graphics. Three different applications are presented in the paper. Sometimes, the problem of finding the shape hull of a given S is conceived as the problem of estimation of planar set in statistical geometry. The present method of quadtree and octree partitioning may be used for the purpose. The principal advantages of these tree techniques are their hierarchical structure, computational economy, and ease of implementation. It is worthwhile investigating their applications in more diverse areas.

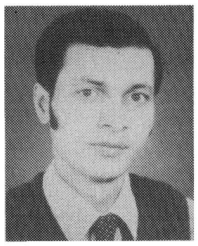
Acknowledgment

Secretarial help rendered by N. Chatterjee, S. De Bhowmik, and S. Chakraborty is gratefully acknowledged.

References

[1] A. Klinger, "Patterns and search statistics," in *Optimizing Methods in Statistics*, J. S. Rustogi, Ed. New York: Academic, 1972.
[2] S. L. Tanimoto and T. Pavlidis, "A hierarchical data structure for image processing," *Comput. Graph. Image Processing*, vol. 4, no. 2, pp. 104-119, 1975.
[3] A. Klinger and C. R. Dyer, "Experiments in picture representation using regular decomposition," *Comput. Graph. Image Processing*, vol. 5, no. 1, pp. 68-105, 1976.
[4] C. L. Jackins and S. L. Tanimoto, "Octtrees and their use in representing three-dimensional objects," *Comput. Graph. Image Processing*, vol. 14, no. 3, pp. 249-270, 1980.
[5] T. Pavlidis, *Algorithms for Grqphics and Image Processing.* New York: Springer-Verlag, 1982.
[6] C. R. Dyer, A. Rosenfeld, and H. Samet, "Region representation: Boundary codes from quadtrees," *Commun. Ass. Comput. Mach.*, vol. 23, no. 3, pp. 171-179, 1980.
[7] R. Ranade, A. Rosenfeld, and J. M. S. Prewitt, "Use of quadtrees for image segmentation," Dep. Comput. Sci., Univ. Maryland, College Park, Tech. Rep. TR-878, 1980.
[8] C. R. Dyer, "Computing the Euler number of an image from its quadtrees," *Comput. Graph. Image Processing*, vol. 13, no. 3, pp. 270-276, 1980.
[9] H. Samet, "Connected component labeling using quadtrees," *J. Ass. Comput. Mach.*, vol. 28, no. 3, pp. 487-501, 1981.
[10] S. Ranade and M. Shneier, "Using quadtrees to smooth images," *IEEE Trans. Syst. Man Cybern.*, vol. SMC-11, no. 5, pp. 373-376, May 1981.
[11] S. Ranade, "Use of quadtrees for edge enhancement," *IEEE Trans. Syst. Man Cybern.*, vol. 11, no. 5, pp. 370-373, May 1981.

[12] E. Kawaguchi and T. Endo, "On a method of binary picture representation and its application to data compression," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-2, no. 1, pp. 27–35, Jan. 1980.
[13] K. Knowlton, "Progressive transmission of grey scale and binary pictures by simple, efficient and lossless encoding schemes," *Proc. IEEE*, vol. 68, no. 7, pp. 885–896, 1980.
[14] H. Samet, "A quadtree medial axis transformation," *Commun. Ass. Comput. Mach.*, vol. 26, no. 9, pp. 680–693, 1983.
[15] R. A. Finkel and J. L. Bentley, "Quadtrees: A data structure for retrieval on composite keys," *Acta Informatica*, vol. 4, no. 1, pp 1–9, 1974.
[16] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Commun. Ass. Comput. Mach.*, vol. 18, no. 9, pp. 509–517, 1975.
[17] A Rosenfeld, H. Samet, C. Shaffer, and R. E. Webber, "Applications of hierarchical data structures to Geographical information systems," Dep. Comput. Sci., Univ. Maryland, College Park, Tech. Rep. TR-1197, June, 1982.
[18] G. T. Toussiant, "Pattern recognition and geometrical complexity," in *Proc. 5th Int. Conf. Pattern Recogn.*, Miami Beach, FL, Dec. 1980, pp. 1324–1347.
[19] R. A. Jarvis, "Computing the shape hull of points in the plane," in *Proc. IEEE Comput. Soc. Conf. Pattern Recogn. Image Processing*, Troy, NY, June 1977, pp. 231–241.
[20] J. Fairfield, "Contoured shape generation—Forms that people see in dot patterns," in *Proc. IEEE Conf. Syst. Man Cybern.*, 1979, pp. 60–64.
[21] M. I. Shamos, "Computational geometry," Ph.D. dissertation, Yale Univ., New Haven, CT, May 1978.
[22] H. Edelsbrunner, D. G. Kirkpatrick, and R. Seidel, "On the shape of a set of points in the plane," *IEEE Trans. Inform. Theory*, vol. IT-29, no. 4, pp. 551–559, July 1983.
[23] M. R. Anderberg, *Cluster Analysis for Applications*. New York: Academic, 1973.
[24] A. Rosenfeld and A. C. Kak, *Digital Picture Processing*. New York: Academic, 1976.
[25] R. F. Ling. "A probability theory of cluster analysis," *J. Amer. Statist. Assoc.*, vol. 68, no. 3, pp. 159–164, 1973.
[26] H. R. Ruspini, "Numerical methods for fuzzy clustering," *Inform. Sci.*, vol. 2, pp. 319–350, 1970.
[27] A. Wald, *Sequential Analysis*. New York: Wiley, 1947.
[28] A. K. Datta, N. R. Ganguli, and S. Ray, "Maximum likelihood methods in vowel recognition—A comparative study," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-4, no. 6, pp. 683–689, Nov. 1982.



B. B. Chaudhuri (S'79–M'80) received the B.Sc. (Hons.), B. Tech., and M. Tech. degrees from Calcutta University, India, in 1969, 1972, and 1974, respectively, and the Ph.D. degree from the Indian Institute of Technology, Kanpur, in 1980.

In 1978 he joined the Indian Statistical Institute as a Faculty member where he is currently an Associate Professor. His research interests include pattern recognition, optical communication, image processing, computer graphics, and artificial intelligence. He has published nearly 60 research papers and co-authored the book *Two-Tone Image Processing and Recognition* to be published by Wiley Eastern Ltd. He is a Referee to some international journals.

Dr. Chaudhuri was awarded the Leverhulme Visiting Fellowship to work at Queen's University, Belfast, Northern Ireland, in 1981 and 1982.