Picture Segmentation by a Tree Traversal Algorithm

STEVEN L. HOROWITZ AND THEODOSIOS PAVLIDIS

Princeton University, Princeton, New Jersey

ABSTRACT. In the past, picture segmentation has been performed by merging small primitive regions or by recursively splitting the whole picture. This paper combines the two approaches with significant increase in processing speed while maintaining small memory requirements. The data structure is described in detail and examples of implementations are given.

KEY WORDS AND PHRASES: picture segmentation, region detection, boundary identification, split-and-merge algorithms, chromosome outlines, human face feature outlines, landscape outlines, radiograph outlines

CR CATEGORIES: 3.63, 5.13, 5.32

1. *Introduction*

An important problem in picture processing, pattern recognition, and scene analysis is the detection of objects from their background. For high contrast and noise free pictures, differentiation can be used successfully for edge detection and then objects can be identified by an edge following algorithm. However, such a simple process fails for noisy pictures with fuzzy boundaries. This is also true when objects are defined by texture rather than brightness level [1, 2]. The major difficulties are the nonuniformity in the brightness level of objects (high frequency noise) and the loss of contrast (low frequency noise). For a given class of picture, one could use a bandpass filter followed by a thresholding operation. However, the design of these operators is nontrivial and may depend critically on the choice of parameters (e.g. the threshold level). This is particularly true for pictures where one must identify many objects, each at a different brightness level. These difficulties have led to the development of alternative techniques, which are usually of one of the following two types.

(1) *Direct region detection.* The algorithm of Brice and Fennema [2, 3] and subsequent extensions of it [4] are typical of this kind. Picture elements (pixels) are merged sequentially to form bigger regions on the basis of global considerations.

(2) *Extended edge detection.* The algorithms of Rosenfeld and Thurston [5, 6], Martelli [7], and Hueckel [8] are examples of this approach. A more complex scheme of differentiation is used in combination with edge following.

In both cases semantic information can be used to improve the results [9, 10].

A generalization of approaches of the first type can be made in terms of functional approximation. If $f(x, y)$ is the brightness function of a picture defined on a domain D, then we may attempt to divide D into the minimum number of regions where $f(x, y)$ satisfies certain constraints (e.g. is approximately constant). Preliminary work has given encouraging results [11] in the sense that one can obtain large regions quickly

This work was supported by the National Science Foundation under Grant GK-36180.

Author's address: Computer Science Laboratory, Department of Electrical Engineering, Princeton University, Princeton, NJ 08540.

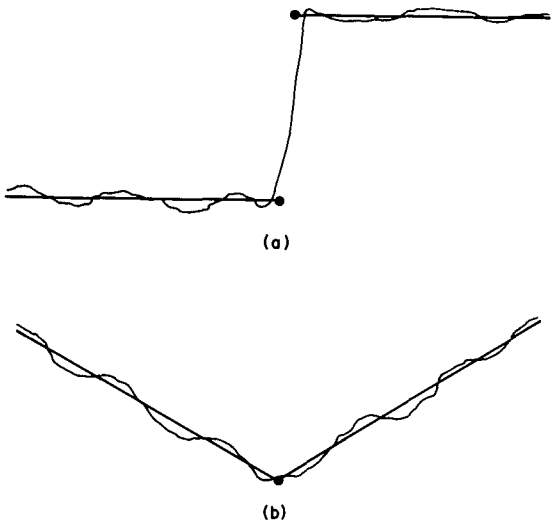


FIG. 1. Illustration of the use of (a) piecewise constant approximations and (b) piecewise linear approximations for segmentation

without the use of semantics, which can be applied later. Two classes of algorithms have been developed. In one, segmentation is first performed along raster lines which are subsequently grouped into bigger regions [11, 12]. In the second, which is the subject of this paper, a direct two-dimensional segmentation is attempted.

In Section 2 we discuss this scheme in the context of other region detection approaches. A detailed comparison with some of the complex edge following schemes [5–8] will be the subject of future work.

As a quick illustration of how functional approximation achieves segmentation, consider the profile of a noisy picture shown in Figure 1. A piecewise constant approximation (a) or a piecewise linear approximation (b) with a variable breakpoint can be used to detect the "edge" between the left part and the right part. The use of such approximations for shape description is not new (see [13, Vol. II, Fig. 10-4]), but it is only recently that fast algorithms applicable to "irregular" data have been developed [14–16].

The effect of such processing is to eliminate high frequency noise without "smearing" boundaries (Figure 1 and also [15, Figs. 3–7]). If a piecewise constant approximation is used then the resulting segmentation usually consists of a few large regions and a large number of small regions corresponding to transition areas. If necessary, these can be merged quickly by a small region elimination routine.

2. *Region Detection Methods*

Region detection methods can be divided into the following two types.

(1) *Merging or bottom-up.* The picture is divided into a large number of small regions (possibly coinciding with single pixels) which are then merged to form larger regions. The "phagocyte" algorithm of Brice and Fennema [3] is typical of this kind. Algorithms which scan a waveform sequentially from left to right to determine the longest interval such that an approximation is below a given tolerance [17] also belong to this class. There have been a number of merging criteria for different types of primitive regions which are discussed in the literature [1–3, 10, 11, 18–20].

(2) *Splitting or top-down.* The picture is successively divided into smaller and smaller regions until certain criteria are satisfied. There were very few applications of this scheme in picture processing until recently [21, 22], but it has been a well-known method for polygonal approximation of boundaries [2].

It is possible to devise a mixed *split-and-merge* scheme, and this has been carried out in the case of waveforms and boundaries with encouraging results [16]. Extension of this strategy is proposed here for pictures. The advantages are (1) an increase in computational speed, and (2) an additional degree of freedom allowing improvement in segmentation quality, both without expanding memory requirements. Below is an abstract formulation of the problem.

Let X be the domain of a picture (e.g. a square). Let $f(x, y)$ be the brightness function defined on X. A logical predicate P is defined on subsets S of X as follows:

$$P(S) = \begin{cases} \text{true if there exists a constant } a \text{ such that } |f(x, y) - a| \leq e \text{ [see ftn. 1]} \\ \quad \text{for all points } (x, y) \in S, \\ \text{false otherwise,} \end{cases}$$

where e is a prescribed error tolerance.

A segmentation of X is a partition of X into subsets S_i, $i = 1, \cdots, m$, for some m such that:

(a) $X = \bigcup_{i=1}^{m} S_i$,

(b) $S_i \cap S_j = \varnothing$ for all $i \neq j$,

(c) $P(S_i) =$ true for all i,

(d) $P(S_i \cup S_j) =$ false for all $i \neq j$,

provided S_i and S_j are adjacent in X. Note that the number m is not unique, nor must it be the minimum under which conditions (a)–(d) hold [23]. In many practical cases, however, imposing these conditions yields an m close to the minimum [16].

A *merging* scheme starts with a partition satisfying (c) and proceeds to fulfill (d); a *splitting* scheme starts with a partition satisfying (d) and proceeds to fulfill (c). A *split-and-merge* procedure begins with an arbitrary partition satisfying neither condition and produces a partition satisfying both.

The above formulation can be expressed in a graph theoretical framework by a tree whose nodes correspond to the subjects of X such that

(i) X is the root of the tree;

(ii) the successors of the node corresponding to a subset S of X are the nodes corresponding to proper, disjoint, collectively exhaustive subsets of S; and

(iii) the leaves of the tree represent the smallest subsets of X under consideration (e.g. single pixels).

Figure 2 shows an example of such a segmentation tree construction. It can be shown that a segmentation corresponds to a node cutset which is the minimal set of nodes separating the root from all the leaves. The root R may be separated from a leaf L in one of three ways:

(i) R is in the cutset.

(ii) L is in the cutset.

(iii) There exists one and only one node N in the cutset such that R is an ancestor of N and L is a descendent of N. This implies that there is no other cutset closer to the root with nodes where $P(S)$ is true. The terms top-down and bottom-up used before can be visualized in terms of moves along the segmentation tree and the corresponding alterations on the original node cutset.

Another operation, known as sidewise merging or *grouping* [11], is accomplished either by transforming the tree structure by combining leaves, or by partitioning the final cutset into region equivalence classes, based on node maxima and minima with respect to the prescribed error tolerance as discussed earlier. Thus adjacent nodes with different predecessors and belonging to different levels may be merged or grouped together to form larger irregular subsets S (possibly of genus greater than zero) provided $P(S)$ holds true. Figure 3 shows an example of grouping and the resulting modifications in tree structure.

[1] Equivalent to $[\max_S (f(x, y)) - \min_S (f(x, y))] \leq 2e$.

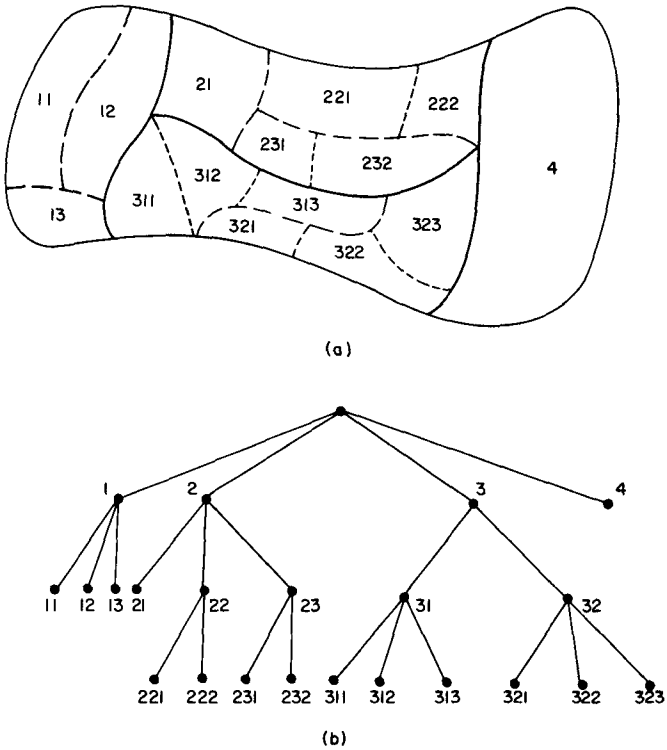


FIG. 2. (a) Example of a directed region segmentation; (b) tree representing the above segmentation

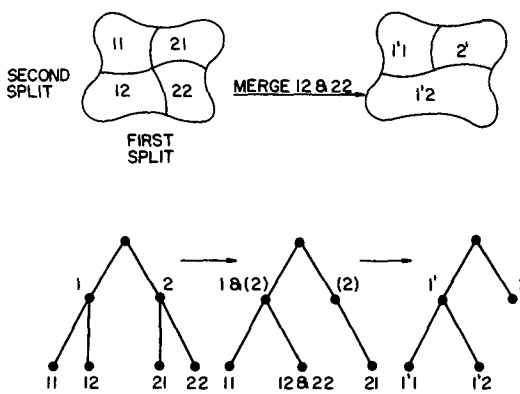


FIG. 3. Illustration of the split-and-merge procedure

It is obvious that this method can be generalized to other than piecewise constant approximations by redefining the predicate $P(S)$. For example, a segmentation by texture can be achieved by replacing the condition $|f(x, y) - a| \leq e$ with

$$\left| f(x, y) - \sum_{u=0}^{m} \sum_{v=0}^{n} F(u, v) e^{j(ux+vy)} \right| \leq e,$$

where m and n are small numbers and $F(u, v)$ is the Fourier Transform ranked in order of significant components.

For simplicity, all the discussion in this paper is limited to the piecewise constant case. However, the algorithm is directly extendable to other criteria and all that is

required is to replace the statements involved directly with the evaluation of $P(S)$. Such statements are flagged in the sequence with the \$ sign. In particular, a comparison to $2e$ should be replaced by an evaluation of $P(S)$, and the evaluation of maxima and minima by whatever calculations are needed for finding an approximation.

The choice of the parameter e is obviously very important. In essence, $2e$ is the tolerated variation of brightness within a single region. Its exact value depends on the class of pictures under consideration and, in particular, the brightness distribution over the scene. Probably the easiest way to estimate it is to try a few different values on a typical sample and then choose the one giving the best results. In general, the precise value is not too important (in contrast, say, to the choice of threshold). Our experience has been that $e = 16$ will give reasonable results for any pictures whose brightness varies from 0 to 127. For other ranges a proportionally scaled value should be used.

3. *Computational Considerations*

In most practical applications the computational efforts involved in splitting and merging are not the same. Merging d subsets into a single set requires two groups of d comparisons (updating the maximum and minimum values of $f(x, y)$ on the domain of the subsets). On the other hand, splitting a single set into d subsets necessitates a reevaluation of the maximum and minimum of $f(x, y)$ on the domain of each of the d subsets. Let

$$\text{cost of merging } \{S_1, \cdots, S_d\} \text{ into } S_0 = C_m,$$

$$\text{cost of splitting } S_0 \text{ into } \{S_1, \cdots, S_d\} = C_s.$$

Note that C_m is independent of $|S_0|$ but dependent on d (i.e. $C_m = 2 \cdot d \cdot \delta$) while C_s is dependent on $|S_0|$ but independent of d (i.e. $C_s = 2 \cdot |S_0| \cdot \delta$), where δ is the cost of a single comparison.

Common as the above model might be, there may be cases where a more symmetrical situation exists, especially in uniform approximation by high order polynomials; also, the dependence on size need not be linear [23].

The choice between a bottom-up or top-down procedure should depend on the level of the tree at which the majority of the nodes in the cutset corresponding to the final segmentation occurs. Asymptotically the top-down procedure is of order $|X| \log_d |X|$ while the bottom-up is of order $|X|$, where d, as before, is also the node degree of the tree. The difference in asymptotic behavior explains the popularity of bottom-up methods in the literature as compared to top-down methods.

Theoretically and experimentally, savings in computation can be achieved if an intermediate level of the tree is chosen as a starting cutset. Such a choice should attempt to minimize the number of splits and merges needed to reach the final cutset, which usually encompasses many levels (some nodes close to the root and others close to or coinciding with leaves) due to the nonuniformity in the distribution of detail over a typical picture. Figure 4(a) illustrates the computational effort yielded by applying the two methods.

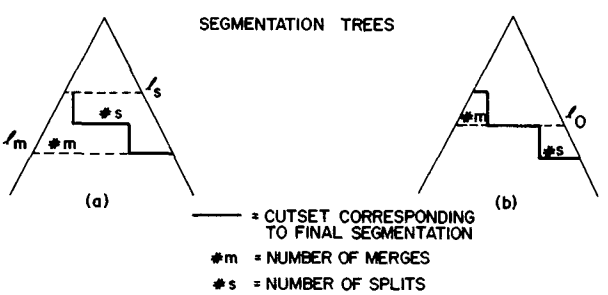


FIG. 4. Illustration of the computational economy achieved by the split-and-merge method

If l_s and l_m are such initial levels for top-down and bottom-up methods respectively, a combined split-and-merge procedure, starting at a middle level l_0 such that $l_m < l_0 < l_s$, can move in both directions with an overall economy as a result. Figure 4(b) illustrates the smaller computational effort associated with a *split-and-merge* approach. That this is indeed the case has been verified for waveforms and boundaries [16, 23], and holds true for the picture examples presented in this paper.

For a given class of pictures a certain distribution of the number and sizes of regions is expected. This should determine the choice of the starting level, i.e. one where the block size is near the average expected size.

4. *Implementation of Split-and-Merge*

In order to process an $N \times N$ picture P (where $N = 2^{l_N}$), the concepts of "segmentation tree" and "node cutset" can be utilized. The segmentation tree is implemented as follows. Each node corresponds to a square picture region segment (block). Leaves of the tree are one-by-one blocks (i.e. single pixels) at level $\log_2 N = l_N$, and all other nodes are defined recursively. A level l node b (where $0 \leq l \leq l_N$) which is located (taken to mean the index of the upper left corner pixel of the corresponding block) at (x, y) has sides of length $z = N/2^l$ and has four successors with sides of length $z/2$ located at (x, y), $(x + z/2, y)$, $(x, y + z/2)$, and $(x + z/2, y + z/2)$. Note that the above definition has the root node (level 0) located at (1, 1). This is, of course, the complete picture with sides of length N. Aside from specifying (x_k, y_k) and z_k, each node b_k has associated values M_k and m_k equal to the maximum and minimum of the brightness function $f(x, y)$ on the corresponding block.

Rather than storing the complete tree in memory (necessitating $5\sum_{i=0}^{l_N} 4^i = 5(4N^2 - 1)/3$ integers), only the cutset is stored as 5 arrays $(x_k, y_k, z_k, M_k, m_k)$ of size N^2. This is achieved by a suitable encoding for the position of b_k in the array given (x_k, y_k) which enables us to retrieve the tree structure, i.e. subsets of four nodes and their common predecessor. Initialization of the array at level l_0 with size of the block sides $s_0 = N/2^{l_0}$ is performed as follows:

```
k ← 0;
DO i = 1 TO N BY s0 ;
  DO j = 1 TO N BY s0 ;
    k ← k + 1; xk ← i; yk ← j; zk ← s0 ;
    Mk ← MAX(p_{i,j} : i = xk , ··· , xk + s0 , j = yk , ··· , yk + s0);   $
    mk ← MIN(p_{i,j} : i = xk , ··· , xk + s0 , j = yk , ··· , yk + s0);   $
    END;
  END,
```

Note that k contains the value $c_0 = 4^{l_0}$ after the initialization step (the number of nodes in the initial cutset).

Now nodes b_{k1}, b_{k2}, b_{k3}, b_{k4} have a common predecessor b_{k0} located at (x_{k0}, y_{k0}) if:

$$k1 = 2^{l_0} \cdot [(x_{k0} - 1) + 2^{l_0}(y_{k0} - 1)]/N + 1,$$

$$k2 = k1 + 2^{l_0 - l}, \quad k3 = k1 + 2^{2l_0 - l}, \quad k4 = k3 + 2^{l_0 - l},$$

for a given level l (where $0 < l \leq l_0$). Figure 5 illustrates the relationship between the five nodes and represents some of the associated variables.

Merging nodes b_{k1}, b_{k2}, b_{k3}, b_{k4} is a process that removes the four nodes from the cutset and replaces them with node b_{k0} (actually b_{k1} updated). In the following listing, s denotes the size of the block sides.

```
l ← l0 ; s ← s0 ;
DO WHILE l > 0;
  DO FOR ALL bk1 ∈ level l;
    IF zk1 = zk2 = zk3 = zk4 = s THEN DO,
```

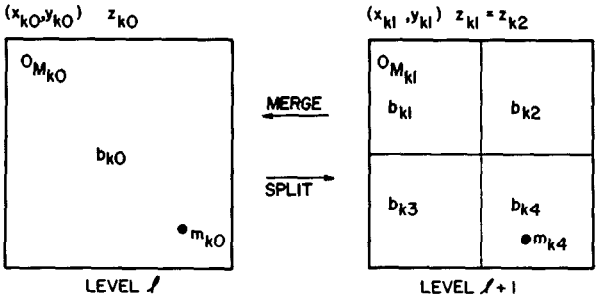


FIG. 5. Illustration of some parameters involved in the splitting or merging of a node

```
    g ← MAX(M_k1, M_k2, M_k3, M_k4);                    $
    h ← MIN(m_k1, m_k2, m_k3, m_k4);                    $
    IF g − h ≤ 2e THEN DO;                              $        [see ftn. 2]
      z_k2 ← 0; z_k3 ← 0; z_k4 ← 0;
      z_k1 ← 2s; M_k1 ← g; m_k1 ← h;                    $
      END;
    END;
  END;
l ← l − 1; s ← 2s;
END;
```

The cutset still has c_0 nodes, but merged nodes either have been updated or have had z set to zero.

Splitting a node b_{k0} (actually b_{k1} before updating) is a process that removes the node from the cutset and replaces it with nodes b_{k1}, b_{k2}, b_{k3}, b_{k4}.

```
c ← c_0 ;
  DO k1 = 1 TO c;
    IF 0 < z_k1 ≤ s_0 THEN DO;
      l ← l_0 ; s ← s_0 ;
      DO WHILE l < l_N ;
        IF M_k1 − m_k1 > 2e THEN DO;                    $        [see ftn. 2]
          k_2 ← c + 1; k_3 ← c + 2; k_4 ← c + 3;
          z_k1 ← s/2; z_k2 ← z_k1 ; z_k3 ← z_k1 ; z_k4 ← z_k1 ;
          x_k2 ← x_k1 ; x_k3 ← x_k1 + z_k1 ; x_k4 ← x_k3 ;
          y_k2 ← y_k1 + z_k1 ; y_k3 ← y_k1 ; y_k4 ← y_k2 ;
        [M_k1 , ··· , M_k4 and m_k1 , ··· , m_k4 are calculated   $
        using values of p_i,j as demonstrated previously],
        l ← l + 1; c ← c + 3
        END;
      ELSE l ← l_N ;
      END;
    END;
  END;
```

This yields the final cutset satisfying properties (a)–(d) in Section 2.

It is easy to see that both processes terminate. Merging can proceed until the cutset consists solely of the root node (e.g. a one-color picture), and splitting can proceed until the cutset consists of the $4^{l_N} = N^2$ leaf nodes only (e.g. an alternating two-color picture). Furthermore, the two processes are mutually exclusive (i.e. all of the merging operations are followed by all of the splitting operations without violating final cutset properties), since once four nodes have been merged they cannot be split. Also, nodes that have been split cannot be merged back to their original configuration, although merging with other adjacent nodes is possible through grouping, which is the third step in the processing sequence.

[2] As defined in Section 2; see ftn. 1.

5. *Implementation of Grouping*

The split-and-merge step is followed by a grouping procedure in order to remove "arbitrary" region boundaries imposed by the "arbitrary" segmentation inherent in the data structure. For example, a natural region which was partitioned originally into two different blocks that are never merged will stay partitioned among the successors of these blocks in the tree. Grouping abandons the tree structure and examines adjacent unrelated blocks found in the final cutset. Block adjacency is obtained from a matrix A initialized as follows:

```
DO k = 1 TO c SUCH THAT z_k ≠ 0;
  DO i = r_k TO x_k + s_k;
    DO j = y_k TO y_k + s_k;
      a_i,j ← k;
      END;
    END;
  END;
```

Thus if a point $p_{i,j}$ in the picture is a member of the block specified by node b_k, the corresponding entry $a_{i,j}$ in the adjacency matrix contains k. Grouping is then executed by a labeling algorithm [24]. A block is either unlabeled and unscanned ($z_k > 0$) or labeled and unscanned ($z_k < 0$ and b_k is on the stack S), or labeled and scanned ($z_k < 0$ and b_k is not on the stack S). Nodes that are labeled during the nth application of the algorithm are placed consecutively on the nth region list R_n. Also the kth block b_k has the value of a new associated variable r_k set to the region number n to which it is assigned:

```
S ← empty; n ← 0;
DO k = 1 TO c;
  IF z_k > 0 THEN DO;
    z_k ← − z_k ; PUSH(b_k) → S; u ← M_k; v ← m_k;
    n ← n + 1, size_n ← 0; sum_n ← 0;
    DO WHILE S¬ empty;
      POP (b_i) ← S; PUSH(b_i) → R_n ; r_i ← n;
      size_n ← size_n + z_i[3] ;
      DO α = x_i TO x_i − z_i − 1;
        DO β = y_i TO y_i − z_i − 1;
          SUM_n ← SUM_n + p_α,β;
          END;
        END;
      DO FOR ALL b_j ∈ A SUCH THAT b_j @ b_i[3];          [see ftn. 3]
        g ← MAX(u, M_j); h ← MIN(v, m_j);          $
        IF g − h ≤ 2e THEN DO;[4]                  $    [see ftn. 4]
          PUSH(b_j) → S; u ← g; v ← h;             $
          END;
      END;
    END;
  END;
END;
```

Note that n contains the final number of regions, size_i contains the number of pixels, and $\text{sum}_i/\text{size}_i$ is the average brightness level of the region specified by R_i. Figure 6 shows an example of the grouping procedure outlined above. The resulting partitioning of the cutset into equivalence classes determining the tree structure transformations is obviously a segmentation satisfying properties (a)–(d) in Section 3.

It should be pointed out that the results of the grouping process are order dependent, since when a block is added to a region the maximum and the minimum of the brightness in that region are updated. This is, of course, a common feature of all techniques

[3] @ = "is adjacent to."

[4] e is as defined in Section 2 (ftn. 2).

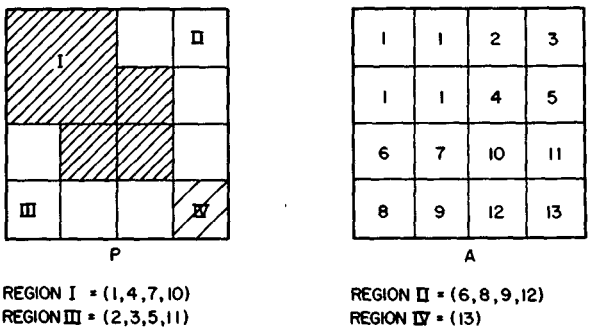


FIG. 6. Illustration of the grouping procedure on a representative picture section with corresponding adjacency matrix

which update the features of a region once new points are added to it. In practice, such order dependence does not create any serious problems (see examples).

After grouping, the picture has been divided into regions where the brightness level remains within the limits specified in Section 3 and adhered to in Section 4. However, such a partition still does not correspond necessarily to a normal visual impression of the picture. There are two reasons for this, both causing the presence of many more regions than "seen." The first is the existence of narrow transition zones between large regions. These zones are not perceived visually as separate entities; instead they are made components of the "real" regions. Second is the existence of high frequency noise. These artifacts are not eliminated by the original approximation using the minimum and maximum brightness level criterion of Section 3. Both problems are well known and usually have been treated on the basis of semantic information [3, 9, 10]. Strong and Rosenfeld have presented a detailed discussion of this problem [25].

In practice, most of these "odd" regions are of very small size (quite often single pixels). This bimodal size distribution [12] suggests that significant improvements can be achieved if small regions are eliminated by combining them with the "nearest neighbor," defined to be the adjacent region with the smallest difference in average brightness. In addition, regions with very similar average brightnesses are merged without regard to size. An average value is used at this stage to overcome the constraints imposed by the minimax value. Also, average brightness is in closer agreement with visual impression and results in higher contrast. For a discussion of various error norms in this context, see [23].

In order to merge adjacent regions, a new data structure providing *region* adjacency information is needed. By utilizing the relations present in the block adjacency matrix A, the region lists R, and the block-region associations r, it is possible to create (in one pass) a region adjacency graph whose nodes correspond to regions and whose branches represent the adjacency relationship. Let n be the number of regions and $\bar{d} = \sum_{i=1}^{n} d_i$, where d_i is the number of regions adjacent to region i (i.e. the degree of graph node i). Since $\bar{d}$ is much smaller than n in practice, the graph is sparse and should be represented in adjacency list form requiring order $n\bar{d}$ storage rather than in other forms requiring order n^2 space [24]. The complete graph need not be stored in memory. Instead, each adjacency list L is generated and accessed only once, thereby using at most $\max_i(d_i)$ locations.

The elimination algorithm proceeds as follows (e_{size} is the largest region to be combined and e_{avg} is the largest difference in average brightness to cause adjacent regions to be merged):

```
DO i = 1 TO n;
   IF size_i > 0 THEN DO;
   DO FOR ALL b_k ∈ R_i ;
     DO FOR ALL b_j ∈ A SUCH THAT b_j @ b_k;
```

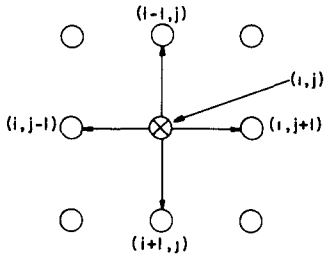


FIG. 7. Elementary search directions during boundary tracing

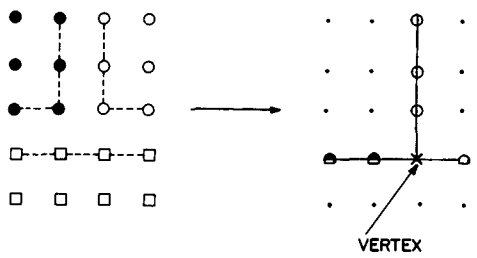


FIG. 8. Production of single-pixel-width boundaries

```
      IF (r_j ≠ i ∧ r_j ∉ L) THEN PUSH (r_j) → L;
      END;
    END;
  avg ← MIN_{l∈L}|sum_l/size_l − sum_i/size_i|,                    $
  m ← INDEX (MIN_{l∈L}|sum_l/size_l − sum_i/size_i|);              $
  DO WHILE (size_i ≤ e_size ∧ avg ≤ e_avg)·
    size_i ← size_i + size_m;                                      $
    sum_i ← sum_i + sum_m;                                         $
    R_i ← R_i||R_m;                                                [see ftn. 5]
    size_m ← 0;
    DO FOR ALL b_k ∈ R_m;
      r_k ← i;
      DO FOR ALL b_j ∈ A SUCH THAT b_j @ b_k;
        IF (r_j ≠ i ∧ r_j ≠ m ∧ r_j ∉ L) THEN PUSH (r_j) → L;
        END;
      avg ← MIN_{l∈L}|sum_l/size_l − sum_i/size_i|;                $
      m ← INDEX (MIN_{l∈L}|sum_l/size_l − sum_i/size_i|,           $
      END;
    END;
  END;
END;
```

6. *Boundary Extraction*

One advantage of picture segmentation by direct partitioning into regions is that closed boundaries are readily determined from the region description. This is in contrast to region formation through edge detection, where extensive tracking is necessary, often yielding contours which are not closed [5–8]. All that is necessary here is a list of regions specified by their upper right-hand corners (x_i, y_i) and a matrix A such that if $p_{i,j}$ belongs to region k then $a_{i,j} = k$.

Beginning with (x_i, y_i), points in the boundary can be extracted by a simple tracing procedure. Only four directions are searched (see Figure 7), thereby enabling a symmetric boundary extension of one unit in two directions in order to avoid having boundaries of two-pixel width (see Figure 8) [12]. This is preferable to the scheme introduced by Brice and Fennema [3], which extends boundaries by $\frac{1}{2}$ unit in 8 directions necessitating a new matrix of $4N^2$.

As the extended boundaries are identified, points where three or four regions meet can be found, also in linear time (see Figure 8). Once these picture "vertices" are determined, the boundaries are easily partitioned and can be approximated by piecewise polynomials for further data compaction or feature analysis [12, 16]. In the present implementation a simple editing has been performed by replacing linear sections of boundaries by their endpoints. This reduces significantly the total number of points P' needed to specify the region boundaries.

[5] || = "is concatenated with."

7. *Examples of Implementation*

The algorithms described in the previous section were implemented on an IBM 360/91 machine using the PL/1 optimizing and Fortran H compilers. Samples were 7-bit, 64×64 pictures ($0 \leq f(x, y) \leq 127$, $l_n = 6$, $n^2 = 4096$). However the actual number of gray levels could be smaller. In order to illustrate the generality of the method, a number of dissimilar subjects were used: human faces, chest x-rays, landscapes, and chromosome pictures. All were digitized using a TV camera connected to an HP 2116 minicomputer. The results are shown in Figures 9–12. In each case, (a) is the original produced by a line printer program [26], (b) is a CALCOMP plot of the boundaries, and (c) is the gray level histogram. Vertices (where three or four regions meet) are marked by little squares. On each original the listed parameters D and L indicate the minimum and maximum of the gray scale used by the line printer program. In each case, D and L were chosen near the extrema of the actual range of gray levels. The symbols used for each level are shown beneath the histogram.

Table I summarizes the statistics of the processing operations. l_0, c_0, and e are as defined in Section 3, e_{avg} and e_{size} are as defined in Section 5, and the rest of the symbols have the following meanings:

- M number of merge operations
- S number of split operations
- B number of nodes in the cutset after the split-and-merge step
- R_i number of regions after the grouping step
- R_f number of regions after the elimination step
- P, P' number of extended boundary points before and after the compaction operation described in Section 6
- V number of vertices as defined in Section 6.

In one example the segmentation is rather simple (Figure 12) and one could achieve comparable results by thresholding. However, the choice of the threshold value is non-trivial [27], while the algorithms proposed require as input only error tolerances such as e, e_{avg}, and e_{size}. This can be chosen and remain invariant for large classes of pictures.

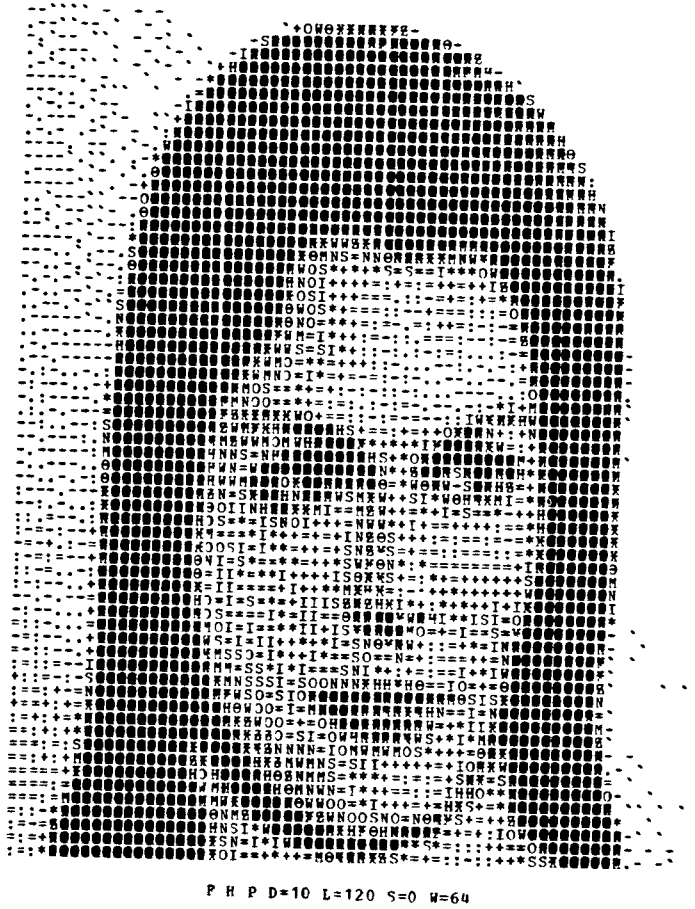
8. *Discussion*

A successful picture segmentation must take into account semantics. However, it is desirable to reduce the amount of data before that step by a general scheme like the one described in this paper. The examples of Section 7 show that one can go quite far in obtaining reasonable segmentations without any significant semantic information. The rather substantial programming effort for the efficient implementation of the initial segmentation algorithm need not be repeated as the subject of interest changes. Only a small number of parameters must be varied. The result is both data compaction and organization. Algorithms of high computational complexity can be used afterward on this reduced description.

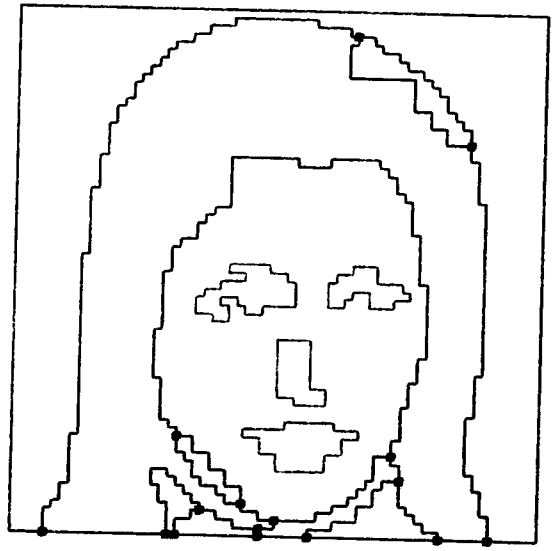
TABLE I. STATISTICS OF SEGMENTATION

Figure (subject)	l_0	c_0	e	e_{avg}	e_{size}	M	S	B	R_i	R_f	P	P'	V	T*
9 (face)	5	1024	16	18	16	164	151	985	148	13	1272	281	16	4.59
10 (X-Ray)	4	256	16	16	16	34	21	217	49	11	1008	62	19	1.45
11 (Scene)	5	1024	9	9	15	164	139	949	186	13	1210	125	23	3.62
12 (Chrom.)	5	1024	12	12	16	223	80	595	115	6	800	192	4	3.12

*picture processing time in seconds (± 10%) - dominated by I/O.



(a)



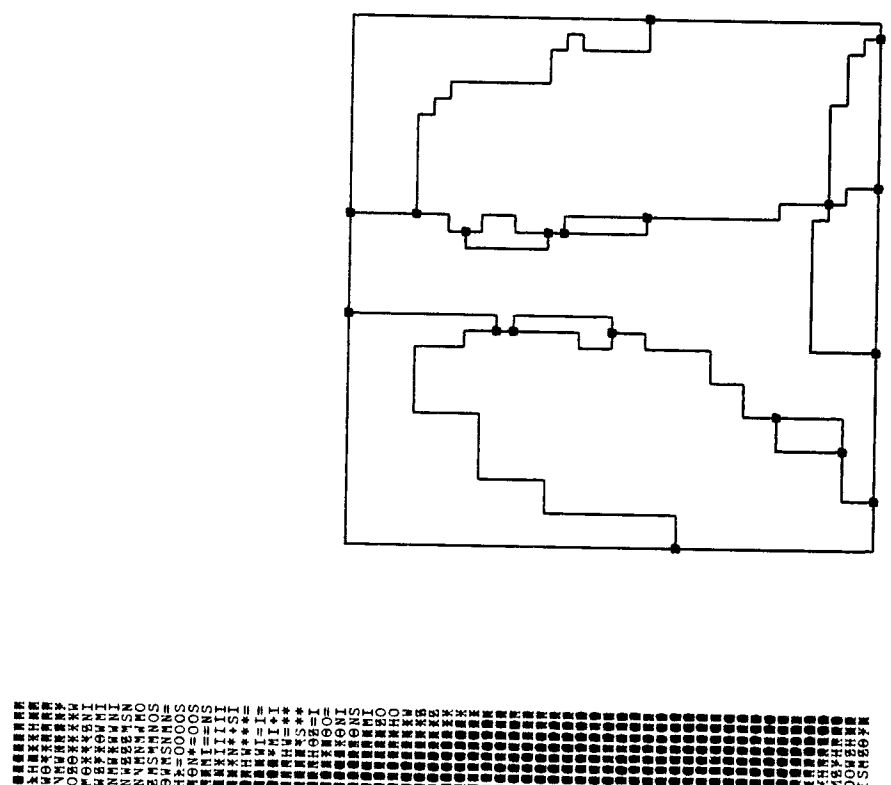
(b)

FIG. 9. (a)–(c) Face digitized from a magazine picture[6]

[6] In all cases there is distortion of the original because the printer prints eight lines per inch and ten symbols per inch per line.

(c)

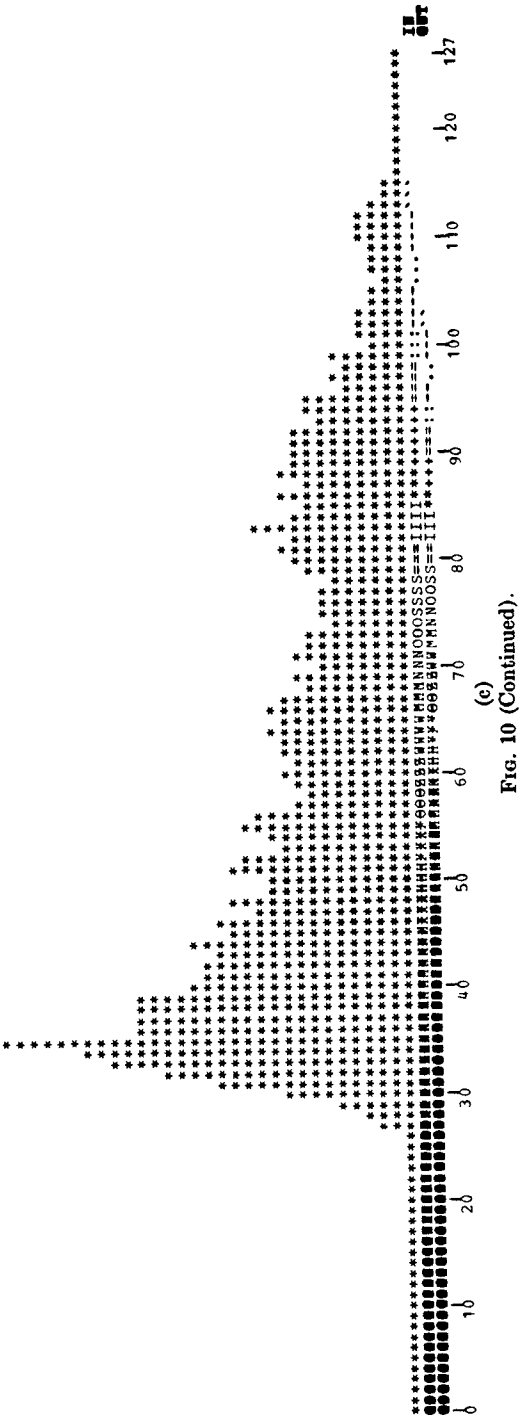
FIG. 9 (Continued).



(a) (b)

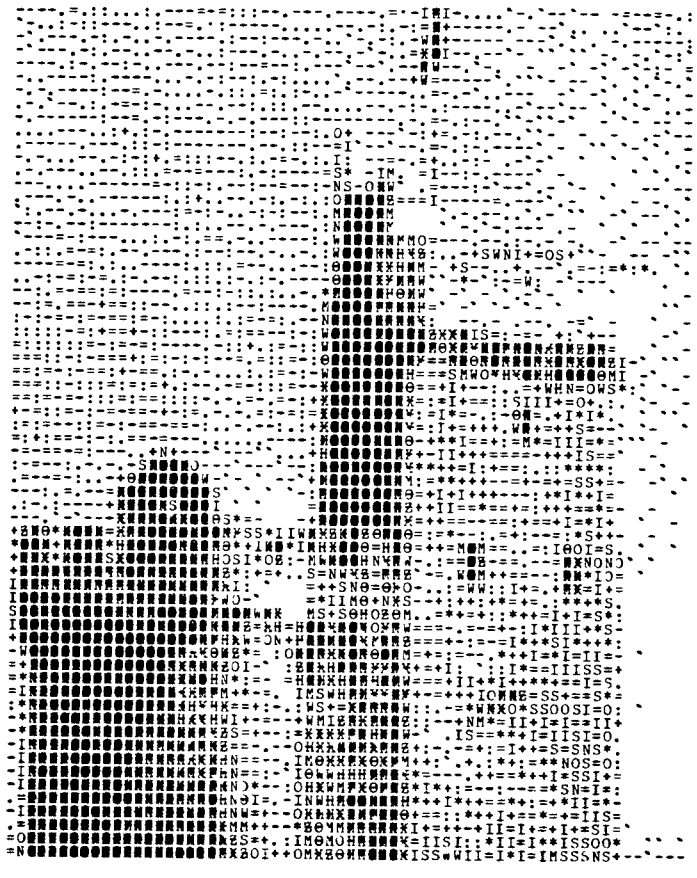
FIG. 10. (a)–(c) Chest X-ray digitized from original film[7]

[7] See footnote 6.



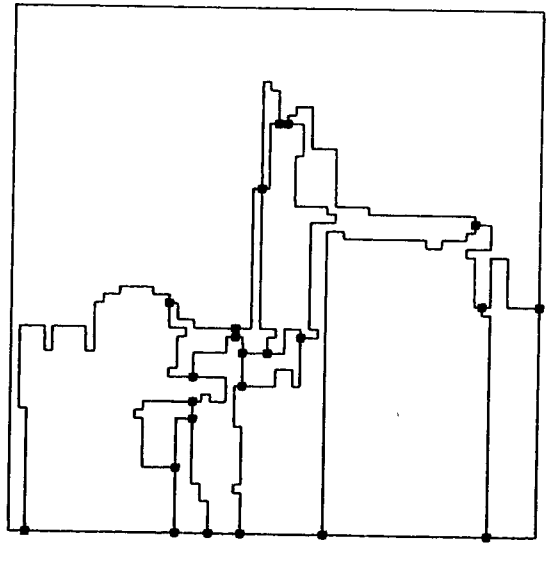
(c)

FIG. 10 (Continued).



F H P D=30 L=110 S=0 W=64

(a)



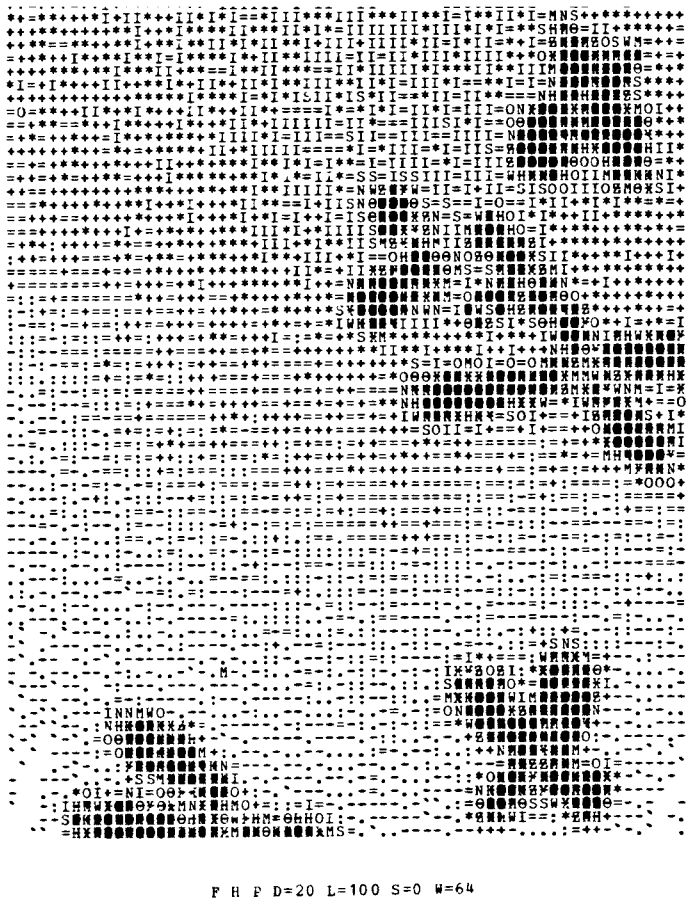
(b)

FIG. 11 (a)–(c) Landscape digitized from a calendar picture (castle)[8]

[8] See footnote 6.

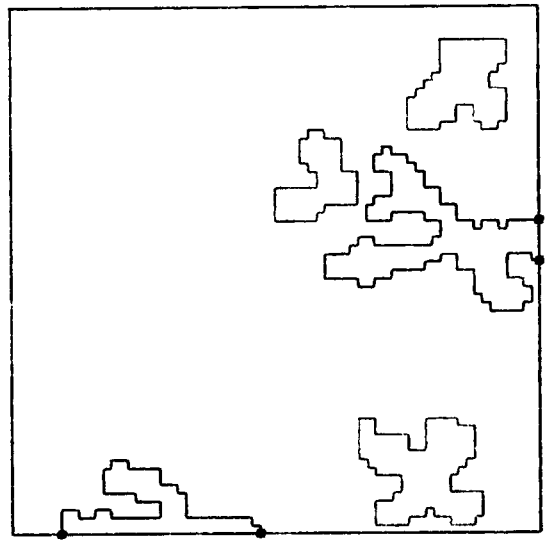
(c)

FIG. 11 (Continued).



F H P D=20 L=100 S=0 W=64

(a)



(b)

FIG. 12. (a)–(c) Chromosome picture digitized from a book illustration[9]

[9] See footnote 6

(e)

FIG. 12 (Continued).

One potential problem in all "global" picture processing schemes is the necessity to keep the whole picture in fast memory during the computations. In this paper there was no problem because the picture resolution was low (64×64). However this usually is not sufficient for many applications, including radiography. The present scheme allows the introduction of "paging" in such cases. Thus a 512×512 picture can be divided into sixty-four 64×64 pictures which can be processed independently up to and including the elimination step. Then the regions so obtained can be used as input to a second pass of the grouping algorithm. Since the number of regions is well below that of pixels, the process can be quite economical in terms of memory requirements.

The results of the segmentation are well suited for syntactic pattern recognition, especially after polygonal or polynomial approximations of the region boundaries [28, 29].

REFERENCES

1. Rosenfeld, A. *Picture Processing by Computer*. Academic Press, New York, 1969.
2. Duda, R.O., and Hart, P.E. *Pattern Classification and Scene Analysis*. Wiley, New York, 1973.
3. Brice, C.R., and Fennema, C.L. Scene analysis using regions *Artif. Intel. J. 1*, 3 (Fall 1970), 205–226.
4. Bajcsy, R. Computer identification of visual surfaces. *Comput. Graphics and Image Processing 2* (1973), 118–130.
5. Rosenfeld, A., and Thurston, M. Edge and curve detection for visual scene analysis. *IEEE Trans. Computers C-20* (May 1971), 562–569.
6. Rosenfeld, A., Thurston, M., and Lee, Y.H. Edge and curve detection: Further experiments. *IEEE Trans. Computers C-21* (July 1972), 677–715.
7. Martelli, A. Edge detection using heuristic methods. *Comput Graphics and Image Processing 1* (1972), 169–182.
8. Hueckel, M.J. A local visual operator which recognizes edges and lines. *J. ACM 20*, 4 (Oct. 1973), 634–647.
9. Kelly, M.D. Edge detection in pictures by computer using planning. In *Machine Intelligence, Vol. 6*, B. Meltzer and D. Michie, Eds., American Elsevier, New York, 1971, pp. 397–409
10. Yakimovsky, Y. Scene analysis using a semantic base for region growing. CS-73-380, Comput. Sci. Dep., Stanford U., Stanford, Calif., June 1973.
11. Pavlidis, T. Segmentation of pictures and maps through functional approximation *Comput. Graphics and Image Processing 1* (1972), 360–372.
12. Feng, H.Y., and Pavlidis, T. The generation of polygonal outlines of objects from grey level pictures. *IEEE Trans. Circuits and Systems CAS-22* (May 1975), 427–439.
13. Rice, J.R. *The Approximation of Functions, Vols. I, II*. Addison-Wesley, Reading, Mass., 1964 and 1969.
14. Pavlidis, T., and Maika, A.P. Uniform piecewise polynomial approximation with variable joints. *J. Approx. Theory 12* (Sept. 1974), 61–69.
15. Pavlidis, T. Waveform segmentation through functional approximation. *IEEE Trans. Computers C-22* (July 1973), 689–697.
16. Pavlidis, T., and Horowitz, S.L. Segmentation of plane curves *IEEE Trans. Computers EC-23* (Aug. 1974), 860–870.
17. Pavlidis, T. Linguistic analysis of waveforms. In *Software Engineering*, J. Tou, Ed., Academic Press, New York, 1971, pp. 203–205.
18. Meisel, W.S. *Computer-Oriented Approaches to Pattern Recognition*. Academic Press, New York, 1972.
19. Strong, J.P., and Rosenfeld, A. A region coloring technique for scene analysis *Comm. ACM 6*, 4 (April 1973), 237–246.
20. Harlow, C.A., and Eisenbeis, S.A. The analysis of radiographic images. *IEEE Trans. Computers C-22* (July 1973), 678–689.
21. Robertson, T.V., Swain, P.H., and Fu, K.S. Multispectral image partitioning. TR-EE 73-26, Purdue U., Lafayette, Ind., Aug. 1973.
22. Klinger, A. Data structures and pattern recognition. Proc. First Int. Joint Conf. on Pattern Recognition, Washington, D.C., Oct. 1973, pp. 497–498 (available from IEEE, New York).
23. Pavlidis, T. The use of algorithms of piecewise approximations for picture processing. Mathematical Software II Conf., Purdue U., Lafayette, Ind., May 1974. Also Tech. Rep. No. 152, Comput. Sci. Lab., Princeton U., Princeton, N.J., May 1974. Submitted to a Technical Journal.

24. STEIGLITZ, K. *An Introduction to Discrete Systems.* Wiley, New York, 1974.
25. STRONG, J.P., AND ROSENFELD, A. A region coloring technique for scene analysis. *Comm. ACM 16*, 4 (April 1973), 237–246.
26. HENDERSON, P., AND TANIMOTO, S. Considerations for efficient picture output via lineprinter. *Comput. Graphics and Image Processing 3* (Dec 1974), 327–335.
27. NAGEL, R.N., AND ROSENFELD, A. Steps towards handwritten signature verification. Proc. First Int. Joint Conf. on Pattern Recognition, Washington, D.C., Oct. 1973, pp. 59–66 (available from IEEE, New York).
28. CHIEN, Y.P., AND FU, K.S. Recognition of X-ray picture patterns. *IEEE Trans. Systems, Man, and Cybernetics SMC-4* (March 1974), 145–156.
29. FENG, H.Y.F., AND PAVLIDIS, T. Decomposition of polygons into simpler components: Feature generation for syntactic pattern recognition. *IEE Trans. Computers C-24* (June 1975), 636–650.

RECEIVED AUGUST 1974; REVISED APRIL 1975