ORNL/TM-13060

RECEIVED
OCT 1 3 1995
OSTI

ornl

OAK RIDGE
NATIONAL
LABORATORY

MARTIN MARIETTA

2-D Image Segmentation Using Minimum Spanning Trees

Ying Xu
Edward C. Uberbacher

MANAGED BY
MARTIN MARIETTA ENERGY SYSTEMS, INC.
FOR THE UNITED STATES
DEPARTMENT OF ENERGY

This report has been reproduced directly from the best available copy.

Available to DOE and DOE contractors from the Office of Scientific and Technical Information, P.O. Box 62, Oak Ridge, TN 37831; prices available from (615) 576-8401, FTS 626-8401.

Available to the public from the National Technical Information Service, U.S. Department of Commerce, 5285 Port Royal Rd., Springfield, VA 22161.

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

DISCLAIMER

Portions of this document may be illegible in electronic image products. Images are produced from the best available original document.

ORNL/TM-13060

Computer Science and Mathematics Division

2-D Image Segmentation Using Minimum Spanning Trees

Ying Xu and Edward C. Uberbacher

DATE PUBLISHED - September 1995

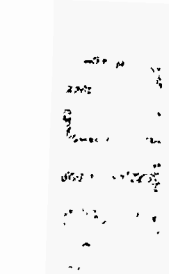
Research supported by the Office of Nonproliferation and National Security
and by the Office of Health and Environmental Research,
U.S. Department of Energy, and the
Laboratory Directed Research and Development Programs

Prepared by the
OAK RIDGE NATIONAL LABORATORY
Oak Ridge, Tennessee 37831
Managed by
LOCKHEED MARTIN ENERGY SYSTEMS, INC.
for the
U.S. DEPARTMENT OF ENERGY
under Contract No. DE-AC05-84OR21400

MASTER

HH

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED



CONTENTS

ABSTRACT

This paper presents a new algorithm for partitioning a gray-level image into connected homogeneous regions. The novelty of this algorithm lies in the fact that by constructing a minimum spanning tree representation of a gray-level image, it reduces a region partitioning problem to a minimum spanning tree partitioning problem, and hence reduces the computational complexity of the region partitioning problem. The tree-partitioning algorithm, in essence, partitions a minimum spanning tree into subtrees, representing different homogeneous regions, by minimizing the sum of variations of gray levels over all subtrees under the constraints that each subtree should have at least a specified number of nodes, and two adjacent subtrees should have significantly different average gray-levels. Two (faster) heuristic implementations are also given for large-scale region partitioning problems. Test results have shown that the segmentation results are satisfactory and insensitive to noise.

1 Introduction

Image segmentation is one of the most fundamental problems in low-level image processing and pattern recognition. The problem is to partition (segment) an image into connected regions of similar textures or similar colors/gray-levels, with adjacent regions having significant dissimilarity. Many algorithms have been proposed to solve this problem[6]. Most of these algorithms can be classified into two classes: (1) boundary detection-based approaches, which partition an image by discovering closed boundary contours, and (2) region growing and clustering-based approaches, which group "similar" (adjacent) pixels into different regions. Region-based approaches can be further classified into two classes, those using global information to do segmentation and those using only local information. Many of the global information-based segmentation algorithms are very computationally time-consuming, while the local information-based approaches are usually sensitive to noise.

In this paper, we present a new region-based segmentation algorithm based on a minimum spanning tree representation of a gray-level image and a tree partitioning algorithm. The algorithm partitions a gray-level image into a number (not pre-determined) of arbitrarily-shaped regions by globally optimizing an objective function, which measures similarities among pixels of each partitioned region, under constraints which prevent from partitioning a homogeneous region into smaller regions. The computational efficiency of the algorithm is the result of a representation of the 2-D image as a tree structure over the pixels of the image, which effectively reduces the region partitioning problem to a simpler tree partitioning problem. Test results have shown that the algorithm is robust in the presence of noise.

The basic idea of the algorithm is as follows. It first builds a weighted planar graph from the given 2-D gray-level image, and then constructs a minimum spanning tree of the graph in such a way that a connected homogeneous region, a region with similar gray levels, corresponds to one subtree of the spanning tree. The tree-partitioning algorithm partitions a tree into a set of subtrees, whose nodes have similar gray levels. In order to prevent the algorithm from forming many small regions or partitioning one homogeneous region into a number of smaller ones, we require that each partitioned region have at least a specified number of pixels and that two adjacent regions have average gray levels that differ by more than a specified value. The algorithm uses dynamic programming to construct an "optimal" segmentation. If we use $\mathcal{I}$ gray levels to represent an image of N pixels the algorithm runs in $O(\max\{(N-K),1\}\mathcal{I}(\log(\mathcal{I})+K^2))$ time and $O(NK\mathcal{I})$ space, where K, $K << N$, is the size of a smallest partitioned region.

The development of this algorithm is a part of the visGRAIL project at Oak Ridge National Lab[10]. visGRAIL is a multi-sensor/neural network based satellite imagery processing/pattern recognition system. The system recognizes areas of significant structure in satellite imagery by recognizing regular features from edge detection and geometric shapes and by combining the features using a neural network system. In visGRAIL, the region segmentation algorithm serves as a front-end of the shape recognition subsystem and also serves

as a robust edge-detection subsystem (using the boundaries of the partitioned regions).

This paper is organized as follows. Section 2 explains the minimum spanning tree representation of a gray-level image. Section 3 presents a tree partitioning algorithm using a dynamic programming approach. Section 4 gives two faster heuristic implementations of the algorithm given in Section 3 for larger images and images with special features. Section 5 describes some applications of the segmentation algorithms.

2 Spanning Tree Representation of an Image

As a powerful graph-theoretic concept, minimum spanning trees have long been used in pattern recognition and classification problems[3,4,9]. In this section, we give a minimum tree representation of a gray-level image.

We assume that an image is given as a $p \times q$ array and each pixel has an integral gray level $\in [0, \mathcal{K}]$, i.e., the whole gray scale $[0,1]$ is divided and discretized into $\mathcal{K}+1$ gray levels. For a given 2-D gray-level image I, we define a weighted planar graph $G(I) = (V, E)$, where the vertex set $V = \{$ all pixels of $I \}$ and the edge set $E = \{(u,v) | u, v \in V$ and $distance(u,v) \leq \sqrt{2} \}$, where $distance(u,v)$ is the Euclidean distance in terms of the number of pixels; each edge $(u,v) \in E$ has a weight $w(u,v) = |\mathcal{G}(u) - \mathcal{G}(v)|$, with $\mathcal{G}(x) \in [0, \mathcal{K}]$ representing the gray level of a pixel $x \in I$. Note that $G(I)$ is a connected graph, i.e., there exists a path between any pair of vertices, and any vertex of $G(I)$ has at most 8 neighbors.

A *spanning tree* T of a connected weighted graph G is a connected subgraph of G such that (1) T contains every vertex of G, and (2) T does not contain cycles. A *minimum* spanning tree is a spanning tree with a minimum total weight.

A minimum spanning tree of a general weighted graph can be found using greedy methods[1], as illustrated by the following strategy: the initial solution is a singleton set containing an edge with the smallest weight, and then the current partial solution is repeatedly expanded by adding the next smallest weighted edge (from the unconsidered edges) under the constraint that no cycles are formed until no more edges can be added. For the above defined planar graph $G(I)$, a minimum spanning tree can be constructed in $O(\|V\| \log(\|V\|))$ time and $O(\|V\|)$ space[1], where $\|\cdot\|$ denotes the cardinality of a set.

We have used Kruskal's algorithm[5] to construct minimum spanning trees. The following gives a pseudo-code of the algorithm. Let the given graph be $G = (V, E)$ and T be the (partial) solution to the minimum spanning tree problem.

$T \leftarrow \emptyset$; $S \leftarrow \emptyset$;
construct a priority queue Q containing all edges of E;
for each vertex $v \in V$ **do** add $\{v\}$ to S;
while $|S| > 1$ **do**
begin
  choose (v,w), an edge in Q of smallest weight;

```
    delete (v, w) from Q;
    if v and w are in different sets s1 and s2 in S then
    begin
       replace s1 and s2 in S by s1 ∪ s2;
       add (v, w) to T;
    end
end
```

The main reason we have used Kruskal's algorithm to construct minimum spanning trees is that the algorithm tends to group pixels of similar gray levels (edges with small weights) together. As a (weak) argument in support of this, we have the following lemma.

Lemma 1 *For a non-negatively weighted connected graph G and its minimum spanning tree T constructed by Kruskal's algorithm, if G_z is the subgraph of G, formed by deleting all non-zero weight edges, and C is a connected component of G_z, then the subgraph of T induced by the vertices of C is connected.*

Proof. This follows directly from the fact that Kruskal's algorithm always chooses the edge of (globally) smallest weight among the remaining candidates as the next edge. □

This lemma implies that if an image contains a number of objects, each of which has a uniform gray level, then the pixels representing each object will form a (connected) subtree of the spanning tree obtained by Kruskal's algorithm.

The following example illustrates the ideas of representing a gray-level image by a minimum spanning tree. Figure 1(a) is a graph representation of an image of a letter "T" on a uniform background with some noise in both the letter and the background. Figure 1(b) is the minimum spanning tree obtained by Kruskal's algorithm. As we can see from Figure 1(b), the letter "T" (slightly deformed) and the background form two (connected) subtrees of the minimum spanning tree (even in the presence of noise) when we cut the spanning tree at "+". It is not difficult to see that a tree partitioning algorithm as outlined in the Introduction Section partitions the image into two regions, the letter "T" and the background, assuming that areas formed by noise are all smaller than the (required) smallest partitioned region.

In general, we would expect that if an image consists of a number of objects represented by homogeneous regions Kruskal's algorithm would build a minimum spanning tree so that each object forms one (connected) subtree. In the following section we present a tree partitioning algorithm that partitions a tree into subtrees based on the similarities of gray levels.

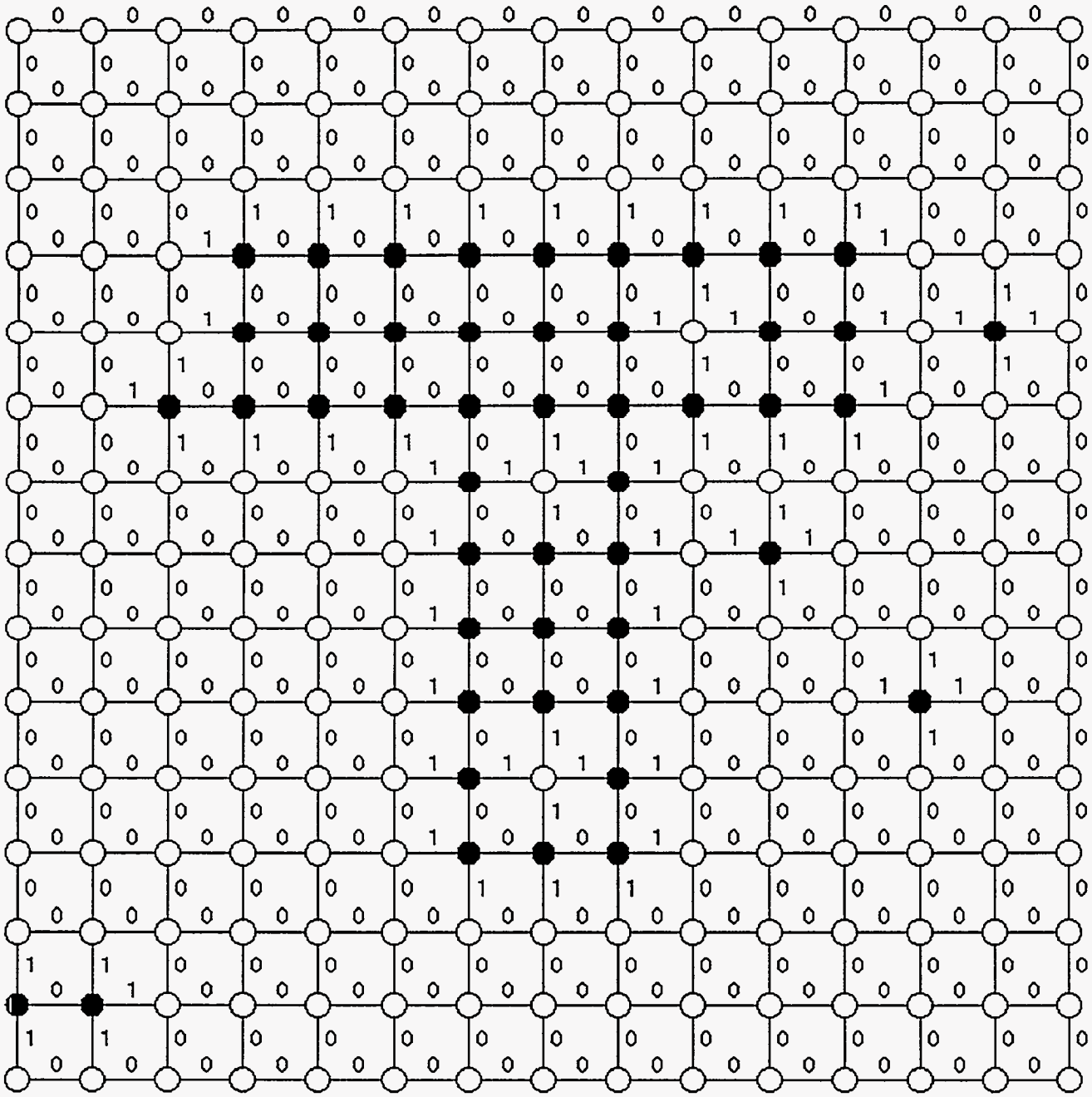


Figure 1: (a) Each circle (node) represents a pixel. Each node has at most four neighbors as represented by the edges (we use four, instead of eight, neighbors for the simplicity of drawing). The number attached to each edge is the weight of the edge. Letter "T" is represented by the solid circles in the middle of the image with some added noise.

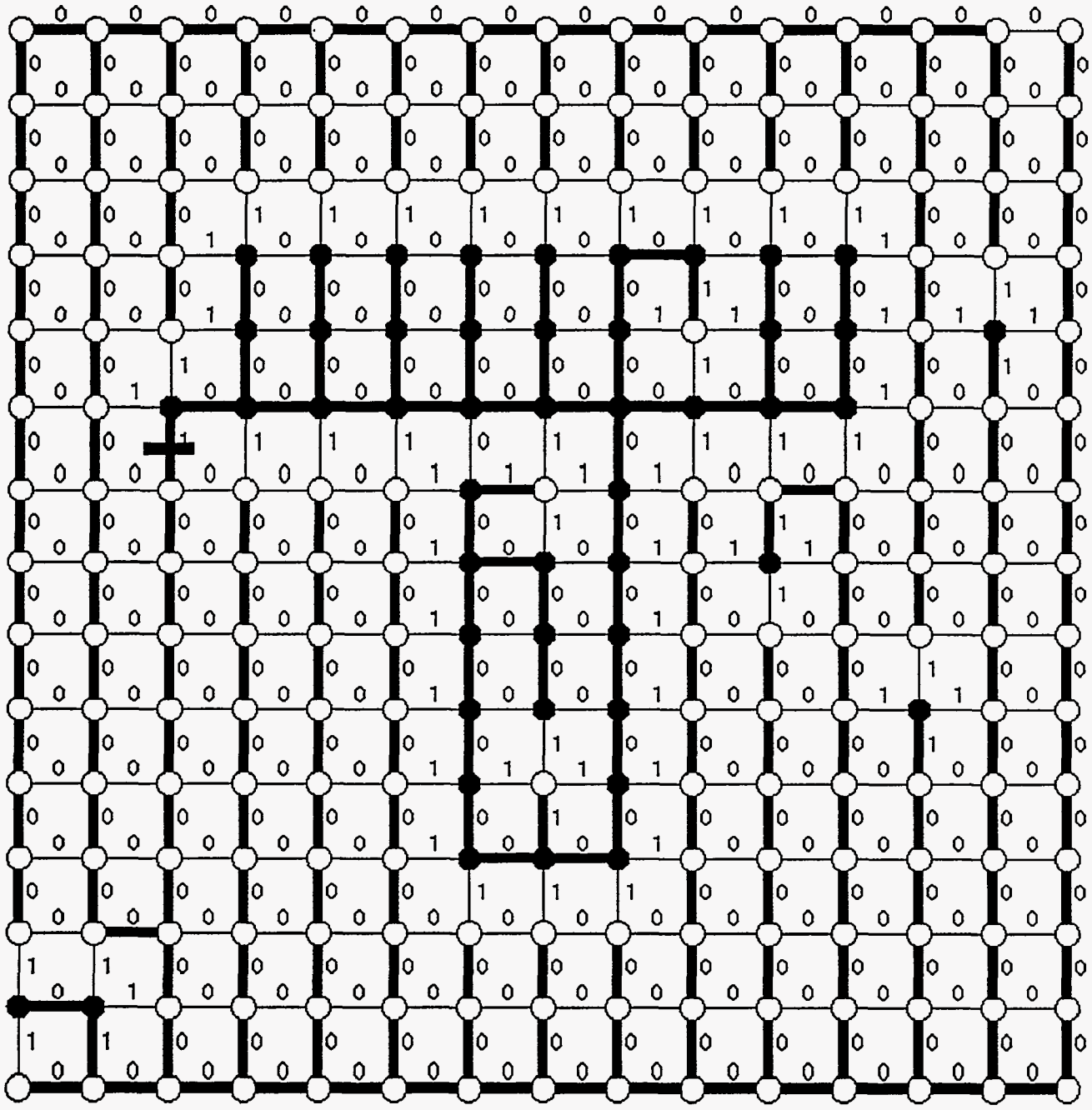


Figure 1: (b) Dark lines represent the minimum spanning tree obtained by Kruskal's algorithm. If the tree is cut at "+" we get the letter "T" and the background as two (connected) subtrees.

3 Tree Segmentation Algorithm

We first give a formal definition of the tree segmentation problem, and then present a dynamic programming algorithm to solve the problem.

3.1 Statement of the problem

Given is a tree T. Each vertex v of T has a gray level $\mathcal{G}(v) \in [0, \mathcal{K}]$. We want to partition T into a set of subtrees T_i, i.e., $T = \bigcup_i T_i$ and $T_i \cap T_j = \emptyset$ for $i \neq j$, such that

$$\begin{array}{lll} \text{minimize} & \sum_i \sum_j (\overline{\mathcal{G}}_i - \mathcal{G}(x_i^j))^2 & \\ \text{subject to:} & (1) & \|T_i\| \geq K, \\ & (2) & |\overline{\mathcal{G}}_i - \overline{\mathcal{G}}_{i'}| \geq D. \end{array}$$

where $\overline{\mathcal{G}}_i$ is the average gray level of T_i, x_i^j is the j^{th} vertex of T_i, T_i and $T_{i'}$ are adjacent subtrees, and K and D are two (application-dependent) parameters, both of which are positive numbers.

One of the difficulties in (efficiently) solving this problem is the need for calculating averages of partitioned subtrees. In this paper we solve a closely related problem formulated as follows, which does not involve calculating averages explicitly. We divide the gray scale into $\mathcal{I}$ equally-sized intervals. Now the problem is redefined to find a partition $T = \bigcup_i T_i$ and an assignment $g_i = g(T_i)$, where g is a mapping from subtrees to the centers of gray-scale intervals, so the following function is minimized.

$$\begin{array}{lll} \text{minimize} & \sum_i \sum_j (g_i - \mathcal{G}(x_i^j))^2 & \\ \text{subject to:} & (1) & \|T_i\| \geq K, \\ & (2) & |g_i - g_{i'}| \geq D. \end{array} \qquad (P)$$

To see the relationship between the two problems consider the following problem. We modify problem (P) by replacing centers of gray-scale intervals by the set of averages of gray levels of all possible subtrees T_i. It can be shown that the modified problem (P) is equivalent to the first problem by simply checking the first and second derivatives of the objective function of the modified (P) on g_i's. So we can consider the (P) is a "discretized" version of the modified problem. As $\mathcal{I}$ gets larger the solution to (P) is approaching to the solution of the first problem as illustrated by the following statements. (1) If T_i is one of the partitioned subtrees in a minimum solution to the problem (P) then $|g_i - \overline{\mathcal{G}}_i| \leq 1/\mathcal{I}$; (2) Let $(g_i, \overline{\mathcal{G}}_i)$ denote the corresponding relation of g_i and $\overline{\mathcal{G}}_i$. If $\{g_i\}$ form a feasible solution (see Section 3.2) to (P) then $\{\overline{\mathcal{G}}_i\}$ form a solution to the first optimization problem after changing its constraint (2) to $|\overline{\mathcal{G}}_i - \overline{\mathcal{G}}_{i'}| \geq D - 1/\mathcal{I}$, and vice versa.

Section 3.2 gives a dynamic programming algorithm to solve the "discretized" tree partitioning problem (P).

3.2 A dynamic programming algorithm

We first define some notation. Any partition $T = \bigcup_i T_i$ with an associated assignment (from subtrees to the centers of gray-scale intervals) $g_i = g(T_i)$ is called a *feasible* solution to the optimization problem (P) if constraints (1) and (2) are satisfied. For any feasible solution, a vertex x_i^j of subtree T_i is said to be *labeled in* g_i if g_i is the center value of the interval assigned to T_i.

The tree-partitioning algorithm first converts a tree into a *rooted* tree by selecting a pixel from one corner of the image (or any other pixel) as the root. Now the *parent-children* relation is well defined. For each tree vertex, the algorithm constructs a minimum solution on the subtree *rooted at* the vertex based on the minimum solutions to the subtrees rooted at its children. It extends a partial solution in such a bottom-up fashion and stops when it reaches the root.

We label the vertices of T consecutively from 1 to $\|T\|$ (the labels are given to tree vertices in any arbitrary order except that 1 is assigned to the root). We use T^i to denote the subtree rooted at vertex i, also referred as subtree T^i. Note the difference between T^i and T_i, the later denoting one of the partitioned subtrees.

Let $score(i, k, g_i)$ denote the minimum value of the objective function among all possible partitions and associated assignments on subtree T^i which satisfy both constraints (1) and (2) except that constraint (1) can be violated for the partitioned subtree containing i, which has at least k, $0 \le k \le K$, vertices and all are labeled in g_i. If we define $scores()$ to be $+\infty$ for all undefined values (note $scores()$ is not defined everywhere by the above definition) we have the following lemma.

Lemma 2 *(a) If $i_1, i_2, ..., i_n$ are the children of vertex i, $n \le 8$, and $1 \le k \le K$, we have*

$$score(i, k, g_i) = \min \sum_{j=1}^{n} score(i_j, k_j, g_i) + (g_i - \mathcal{G}(i))^2, \text{ for } k = \sum_{j=1}^{n} k_j, k_j \ge 0, \quad \|T^i\| \ge K$$

$$scores(i, k, g_i) = \begin{cases} \sum_{p \in \mathcal{D}(i)} (g_i - \mathcal{G}(p))^2 & \|T^i\| = k \\ +\infty & \|T^i\| \ne k \end{cases} \qquad \|T^i\| < K$$

where $\mathcal{D}(i)$ is the set of all i's descendants, i is defined to be $\in \mathcal{D}(i)$ and $score(i_j, 0, g_i)$ is defined to be

$$\min_{|g_i' - g_i| \ge D} score(i_j, K, g_i').$$

(b) $\min_g score(1, K, g)$ is a minimum solution of (P), where 1 represents the tree root.

Proof. The essence of this lemma is the recursive relation of $scores()$'s on a tree vertex and its children. This can be checked by an inductive argument on the number of children. □

Based on Lemma 1, we can solve the optimization problem (P) by calculating $score()$ for each tree vertex in a bottom-up fashion using the recurrence from Lemma 1(a), and stopping at the tree root. To get the tree partition corresponding to $\min_g score(1, K, g)$, some simple bookkeeping needs to be done while calculating $score()$'s.

The pseudo-code to calculate *score*()'s is given below. We omit further details on the bookkeeping for recovering the tree partition. The input to the algorithm is the minimum spanning tree T obtained by Kruskal's algorithm. To efficiently calculate the recurrence of Lemma 1(a), we need to introduce an auxiliary quantity $score_{temp}(i,k,g)$ for each i,k,g. For the simplicity of discussion, in the following we use $g \in [1,\mathcal{I}]$ to denote that g is the center of one of the gray intervals from 1 through $\mathcal{I}$.

Procedure *cal_scores*(i)

1. **if** vertex i has more than K descendants **then**
2. **begin**
3. **for** each of the n children, i_j, of i **do** **call** *cal_scores*(i_j);
4. set $score_{temp}(i,k,g) \leftarrow 0$, for all $k \in [1,K]$, and $g \in [1,\mathcal{I}]$;
5. **for** $j = 1$ **step** $+1$ **until** n **do**
6. **for** $g = 1$ **step** $+1$ **until** $\mathcal{I}$ **do**
7. **for** $k = 1$ **step** $+1$ **until** K **do**
8. $score_{temp}(i,k,g) \leftarrow \min\{score_{temp}(i,k_1,g) + score(i_j,k_2,g)\}$, for $k = k_1 + k_2, k_1, k_2 \geq 0$;
9. **for** $k = 1$ **step** $+1$ **until** K **do**
10. $score(i,k,g) \leftarrow \min_{j \geq k}\{score_{temp}(i,j,g) + (g - \mathcal{G}(i))^2\}$;
11. **for** $g = 1$ **step** $+1$ **until** $\mathcal{I}$ **do**
12. set $score(i,0,g) \leftarrow \min score(i,K,g')$, for $|g' - g| \geq D, g' \in [1,\mathcal{I}]$.
13. **end**
14. **else**
15. **begin**
16. **for** $g = 1$ **step** $+1$ **until** $\mathcal{I}$ **do**
17. **for** $k = 1$ **step** $+1$ **until** K **do**
18. **if** $k = \|T^i\|$ **then** set $score(i,k,g) \leftarrow \sum_{p \in \mathcal{D}(i)}(g - \mathcal{G}(p))^2$;
19. **else** set $score(i,k,g) \leftarrow +\infty$.
20. **end**

Theorem 1 shows the efficiency of our tree partitioning algorithm.

Theorem 1 $\min_g score(1,K,g)$ *can be correctly calculated by the above algorithm in* $O(\max\{(\|T\| - K), 1\}\mathcal{I}(\log(\mathcal{I}) + K^2))$ *time and* $O((\|T\|\mathcal{I}K)$ *space.*

Proof. The correctness of the algorithm can be checked easily based on Lemma 1.

It takes $O(\|T\|)$ time to calculate the numbers of descendants for all the tree vertices using a post-order traversal on the tree. For each tree vertex, Line 4 takes $O(\mathcal{I}K)$ time and space; Lines 5 - 8 take $O(K^2\mathcal{I})$ time and $O(\mathcal{I}K)$ space; Lines 9 - 10 take $O(K + \mathcal{I})$ time and space; and Lines 16 - 19 take $O(K\mathcal{I})$ time and space. To calculate Lines 11 - 12 efficiently we can use a *search tree* data structure[8] to represent the list $\{score(i,0,1)$, ...,

$score(i, 0, \mathcal{I})\}$. So Line 12 can be calculated using two search tree operations, which takes $O(\log(\mathcal{I}))$ time and constant space.

To initialize the search tree data structure takes $O(\mathcal{I})$ time and space. Lines 2 - 13 are executed $O(\|T\| - K)$ times and Lines 15 - 19 are executed $O(\|T\|)$ times. Hence the theorem follows. □

In many applications, $\mathcal{I}$ is at most 256, and hence can be considered as a constant. So the computational resources used are $O(\max\{(\|T\| - K), 1\}K^2\})$ in time[1] and $O(\|T\|K)$ in space.

4 Heuristic Implementations

The algorithm presented in Section 3 works effectively for images of moderate sizes, but when images get larger, perhaps 1000x1000 pixels or above, the computational time and memory required can become too large for practical applications. This section gives two faster heuristic implementations of the algorithm of Section 3, which avoid using K, the smallest region size, as a parameter.

4.1 Heuristic # 1: using penalty functions

One way to avoid using the amount of memory and time required for the above algorithm is to use penalty functions to approximately realize the requirement of each subtree having at least K vertices, i.e., we penalize each time a division needs to be done. Instead of minimizing the above objective function we solve the following problem. Partition T into $\bigcup_{i=1} T_i$ and find assignments g_i to each T_i to minimize the following function.

$$\begin{array}{ll} \text{minimize} & \sum_i(\sum_j(g_i - \mathcal{G}(i))^2 + penalty\_factor) \\ \text{subject to:} & |g_i - g_{i'}| \geq D. \end{array} \qquad (P')$$

where $penalty\_factor$ is a fixed positive integer, which can be adjusted based on the practical applications.

[1]This time bound can be further improved to $O(\max\{(\|T\| - K), 1\}K\log(K)))$ using a theoretically faster algorithm to solve the following convolution problem. Let A and B be defined as

$$A = \begin{pmatrix} a_0 \\ a_1 \\ . \\ a_{n-1} \end{pmatrix}, \quad B = \begin{pmatrix} b_0 \\ b_1 \\ . \\ b_{n-1} \end{pmatrix},$$

We want to calculate C, with each of C's element c_i, $i \in [0, n-1]$, defined as

$$c_i = \min_{i=j+k, j,k\geq 0} a_j + b_k.$$

Using a technique similar to Fast Fourier transform (FFT), this problem can be solved in $O(n\log(n))$ time[7].

This problem can be solved as follows. Define $score(i, g)$ to be the minimum score of the objective function of (P' among all feasible solutions over subtree T^i under the constraint that the vertex i is labeled in g. In the following, we omit discussions on cases where $\|T^i\| < K$.

For each tree vertex i, and its children $i_1, i_2, ..., i_n$, we can show the following.

$$score(i, g) = \sum_{j=1}^{n} \min\{score(i_j, g), \min_{|g-g'|\geq D} score(i_j, g') + penalty\_factor\} + (g - \mathcal{G}(i))^2,$$

and $\min_g score(1, g)$ corresponds to the minimum solution of (P'). By a similar discussion to the one in Theorem 1, we have the following result.

Theorem 2 *The tree partitioning problem using the penalty-function heuristic can be solved in $O(\|T\|\mathcal{I}\log(\mathcal{I}))$ time and $O(\|T\|\mathcal{I})$ space.* □

4.2 Heuristic # 2: linearization of a spanning tree

Another heuristic we have used for images with special features is to first "linearize" the tree T and then solve a one-dimensional list partitioning problem. Since some relational information among pixels may be lost during the linearization this strategy only works well when the given image is structurely not very complicated. For example, it works very effectively when an image has a number of objects in the same background and no objects contains other objects.

The linearization of a tree is achieved by constructing a mapping from the tree vertices to a linear list. We achieve this by first converting the tree into a rooted tree as before, and then labeling each tree vertex using the depth-first search order from 1 to $\|T\|$. By ordering the tree vertices in the increasing order of their mappings, we get a linearization of the tree (in the following we still use T to represent the mapped list). A useful property of this mapping is that when the image is not very complicated each subtree representing an object corresponds a very small number of sub-intervals of $[1, \|T\|]$. In the ideal situation, each object only corresponds to one sub-interval and the background may be divided into a number of sub-intervals.

When this premise holds for a practical application the following algorithm for one-dimensional path segmentation has proven to be effective.

Now we give the formulation of the 1-D list segmentation problem. We want to partition $T = \bigcup T_i$ and find an assignment $g_i = g(T_i)$, so the following function is minimized.

$$\begin{array}{lll} \text{minimize} & \sum_i \sum_j (g_i - \mathcal{G}(i))^2 & \\ & & (P'') \\ \text{subject to:} & (1) \quad \|T_i\| \geq K, & \\ & (2) \quad |g_i - g_i'| \geq D. & \end{array}$$

Let $score(i, g, s)$ denote the minimum score of the objective function of (P'') among all feasible solutions on the sublist T^i which satisfy both constraints (1) and (2) except

constraint (1) can be violated for the partitioned sublist containing i when $s = 0$, s is either 0 or 1. We can show the following.

$$score(i, g, 0) = \min_{|g'-g| \geq D} \{score(i-1, g, 0), score(i-1, g', 1)\} + (g - \mathcal{G}(i))^2,$$

$$score(i, g, 1) = \min_{|g'-g| \geq D} \{score(i-K, g', 0) + \sum_{j=0}^{K-1} (g' - \mathcal{G}(i-j))^2\},$$

and $\min_{g,s} score(1, g, s)$ corresponds to the minimum solution of (P''). We have the following.

Theorem 3 *The tree partition problem can be solved using the linearization heuristic in* $O(\|T\|\mathcal{I} \log(\mathcal{I}))$ *time and* $O(\|T\|\mathcal{I})$ *space.* □

5 Applications and Discussions

We have implemented the algorithms presented in the previous sections in the visGRAIL system to process satellite imageries and extract features for high-level pattern recognition. In one application of the minimum spanning tree representation and the tree partitioning algorithms we extracted geometric shape information from different regions (objects). In another application we used the region segmentation algorithms to detect edge lines. We have combined traditional edge detection algorithms[2] and the region segmentation result to obtained more reliable edge lines. A simple way for such an application is to use a pixel-by-pixel "and" operation on the region boundaries and edges detected by edge detection algorithms after some dilation/erosion operations[4]. The rationale is that edges obtained by edge detection algorithms represent pixels in gray-level local optima and region boundaries represent delineation lines of regions while true edge lines should satisfy both criteria.

While the heuristic implementation using linearization is the weakest algorithm the performances of the other two algorithms are comparable. In cases with no prior knowledge about the minimum size of an object to be recognized, the heuristic implementation using penalty functions may even give better segmentation results. In general, the segmentation performance of these two algorithms are very adequate for the application of visGRAIL to date.

The Figure 2 gives an example of image segmentation by the heuristic algorithm given in Section 4.1.

Some research work is under way to study the limitations of our image segmentation algorithm under different types of noise. In the following we show two test cases. In these tests, we used an artificially-made example consisting of two letters, "T" and "E", in a uniform background. Both letters are in gray level 100 and the background is in gray level 150 (on a 0 to 255 scale). Two types of noise are added separately to the given image and the segmentation results in the presence of noise are shown.

In the first test, we use the following model to generate noise. Each pixel of the given image has probability $\mathcal{P}$ to keep its current gray level and probability 1 - $\mathcal{P}$ to randomly take a gray level $\in [0, 255]$. Figures 3 and 4 show the results for $\mathcal{P} = 0.65$ and $\mathcal{P} = 0.3$, respectively.

The second test shows the segmentation results in the presence of additive Gaussian noise. We add to each pixel a random integral value according to the censored normal distribution $N(0, \sigma^2)$ to [-128, 128]. Figures 5 and 6 give the test results for $\sigma = 40$ and $\sigma = 50$, respectively, after a preprocessing of averaging (we average the gray levels of each pixel and its 8 neighbors).

Computational efficiency is gained by reducing a region partitioning problem to a tree partitioning problem. But we may have to pay for this gain by lossing some (region) partitioning accuracy in some very noisy imagery, as indicated by some preliminary tests we have conducted. Part of the reason for lossing some accuracy is due to the "incorrect" configuration of the minimum spanning tree constructed, e.g., an "object" may be represented by more than one (not connected) subtrees due to noise as illustrated in Figure 6. Our current scheme in assigning weight to each edge uses only local information, which is vulnerable to noise. Some study is currently under way to make the algorithm even less sensitive to noise in the level of tree construction.

6 Conclusion

We have proposed and implemented an effective and efficient 2-D image segmentation algorithm. Additional theoretical and empirical analysis is underway to assess limitations of the algorithm in the presence of noisy imagery. Although the algorithm is presented to solve 2-D image segmentation problems it can be applied to 3-D gray level images without increasing any of the asymptotic complexity results.

Acknowledgements

This research was supported by the United States Department of Energy, under contract DE-AC05-84OR21400 with Martin Marietta Energy Systems, Inc.

The authors would like to thank Dr. Reinhold C. Mann and Dr. Victor Olman for many helpful discussions related to the work presented in this paper and for their critical review of the manuscript. The authors also want to express their great appreciation to Ron Lee for answering their endless software/system questions during the implementation of the algorithms.

References

[1] AHU A. V. Aho, J. E. Hopcroft, and J. D. Ullman (1974), *The Design and Analysis of Computer Algorithms*, Readings, MA, Addison-Wesley Publishing Company.

[2] J. F. Canny (1986), "A Computational Approach to Edge Detection", *IEEE Transaction on Pattern Analysis and Machine Intelligence*, Vol. PAMI-8, No. 6, pp. 679 - 698.

[3] R. O. Duda and P. E. Hart (1973), *Pattern Classification and Scene Analysis*, New York, John Wiley and Sons.

[4] R. C. Gonzalez and P. Wintz (1987), *Digital Image Processing (second edition)*, Readings, MA, Addison-Wesley Publishing Company.

[5] J. B. Jr. Kruskal (1956), "On the shortest spanning subtree of a graph and the traveling salesman problem", *Proc. Amer. Math Soc*, Vol. 7, No. 1, pp. 48 - 50.

[6] N. Pal and S. Pal (1993), "A review on image segmentation techniques", *Pattern Recognition*, Vol. 26, No. 9, pp. 1277 - 1294.

[7] N. S. V. Rao (1995), Private communication.

[8] R. E. Tarjan (1983), *Data Structures and Network Algorithms*, Philadelphia, PA, Society for Industrial and Applied Mathematics Press.

[9] G. Toussaint (ed.) (1985), *Computational Geometry*, North-Holland, Elsevier Science Publishers.

[10] E. C. Uberbacher, Y. Xu, M. Beckerman, C. Glover, R. Lee and R. C. Mann (1995), "Analysis of Satellite Imagery Using Multi-sensors/Neural Network Systems", Submitted to *Applied Imagery Pattern Recognition 95'*.

Figure 2: (a) is a 496x494 gray level image.

Figure 2: (b) is the result of image segmentation.

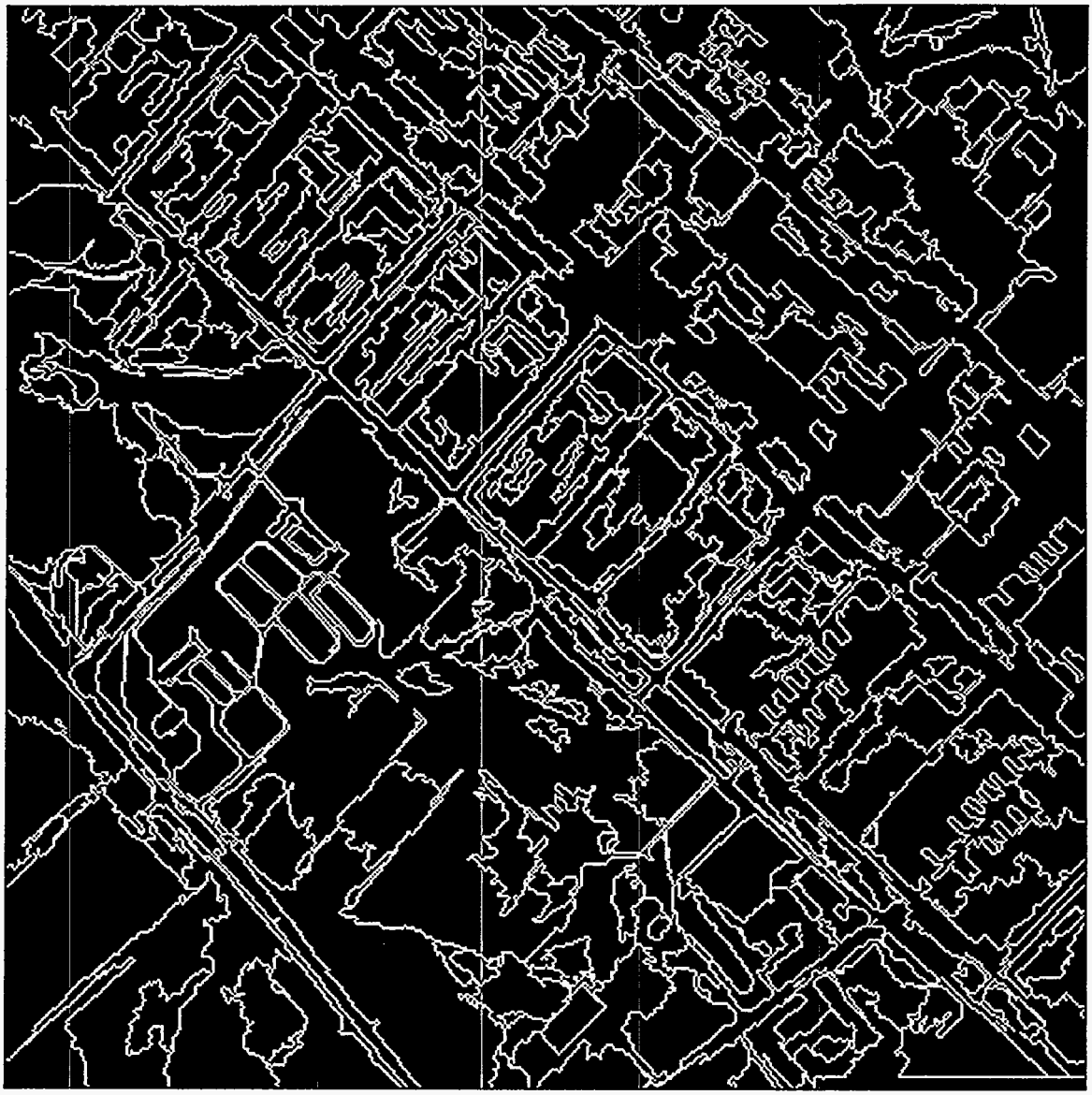


Figure 2: (c) shows the boundaries of the segmented regions.

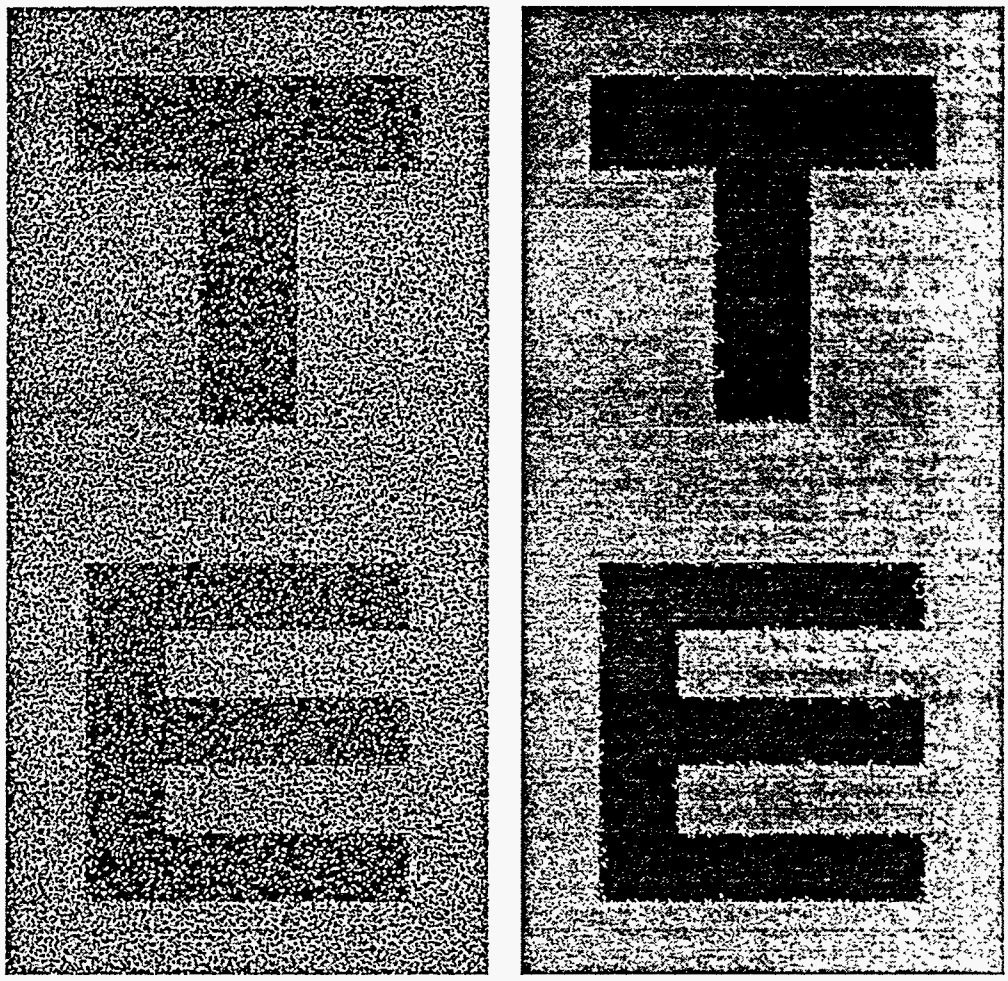


Figure 3: The image with noise and segmentation results for $\mathcal{P} = 0.65$.

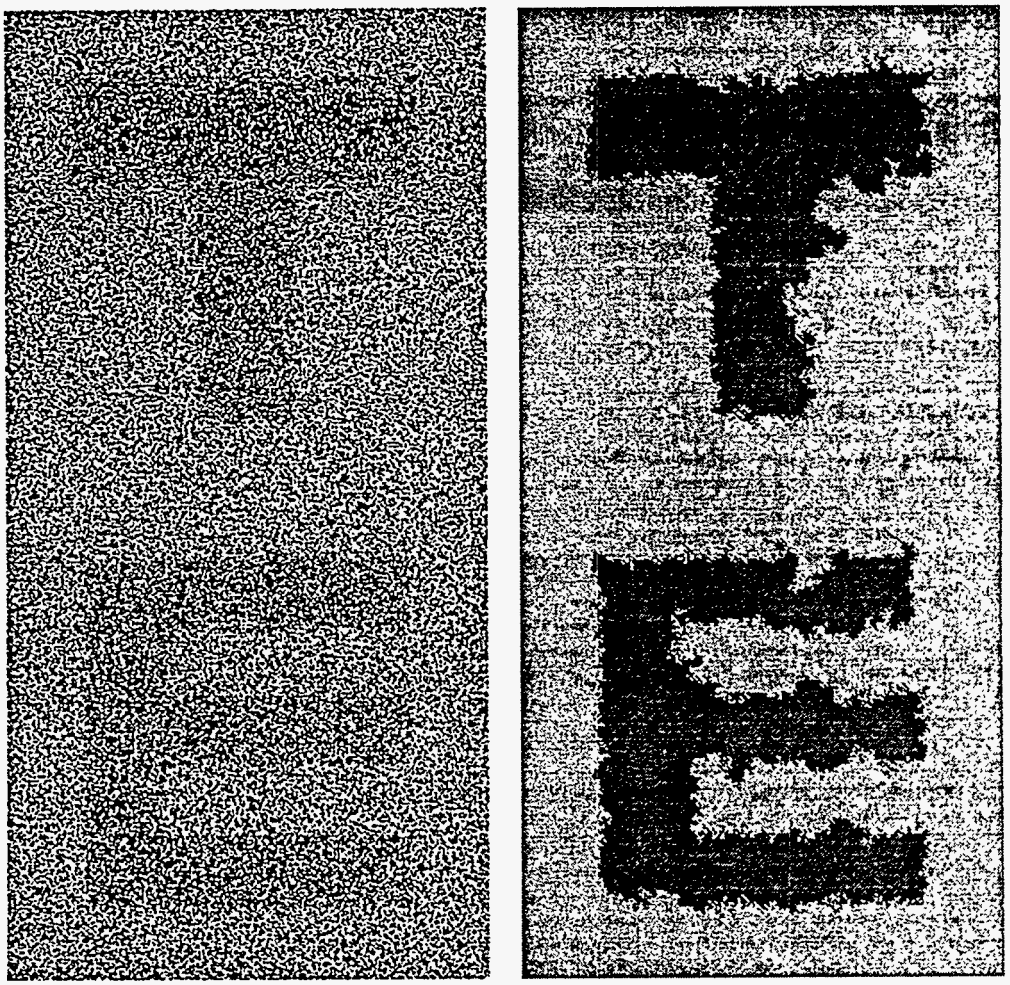


Figure 4: The image with noise and segmentation results for $\mathcal{P} = 0.3$.

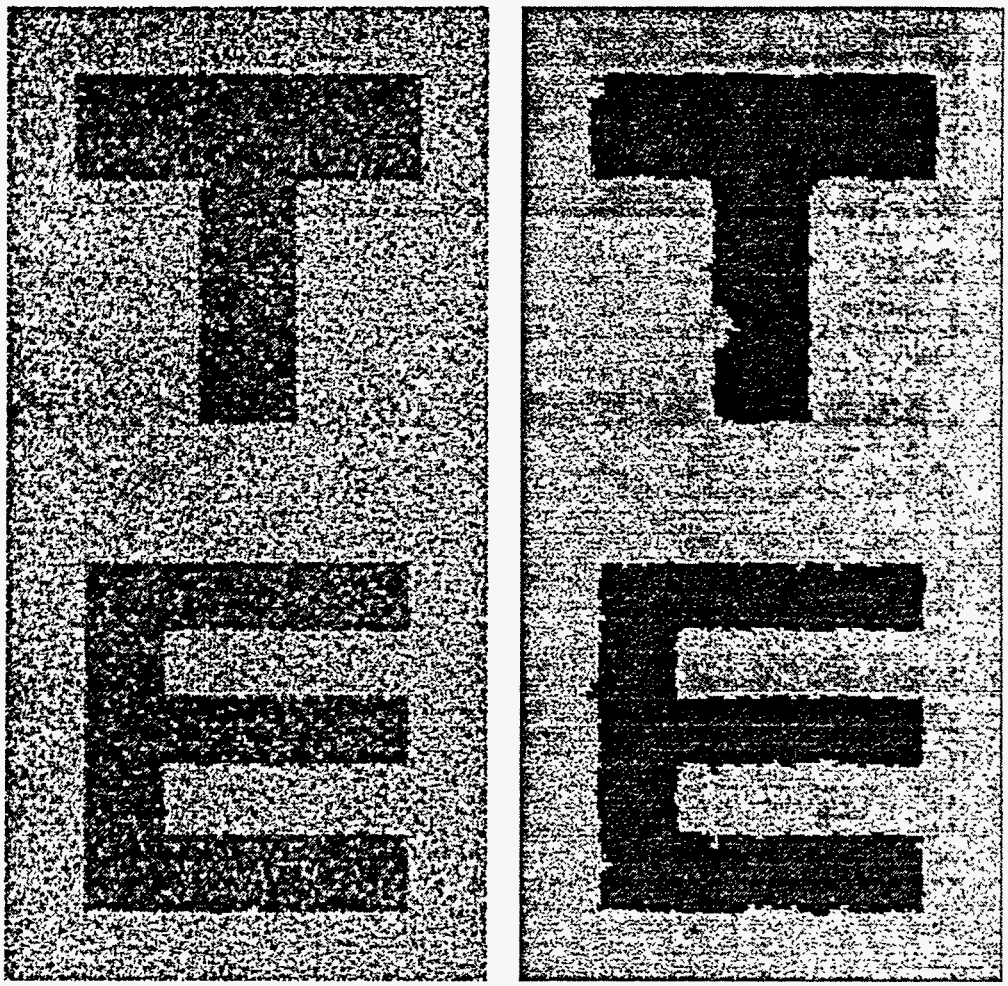


Figure 5: The averaged noisy image and segmentation results for $\sigma = 40$.

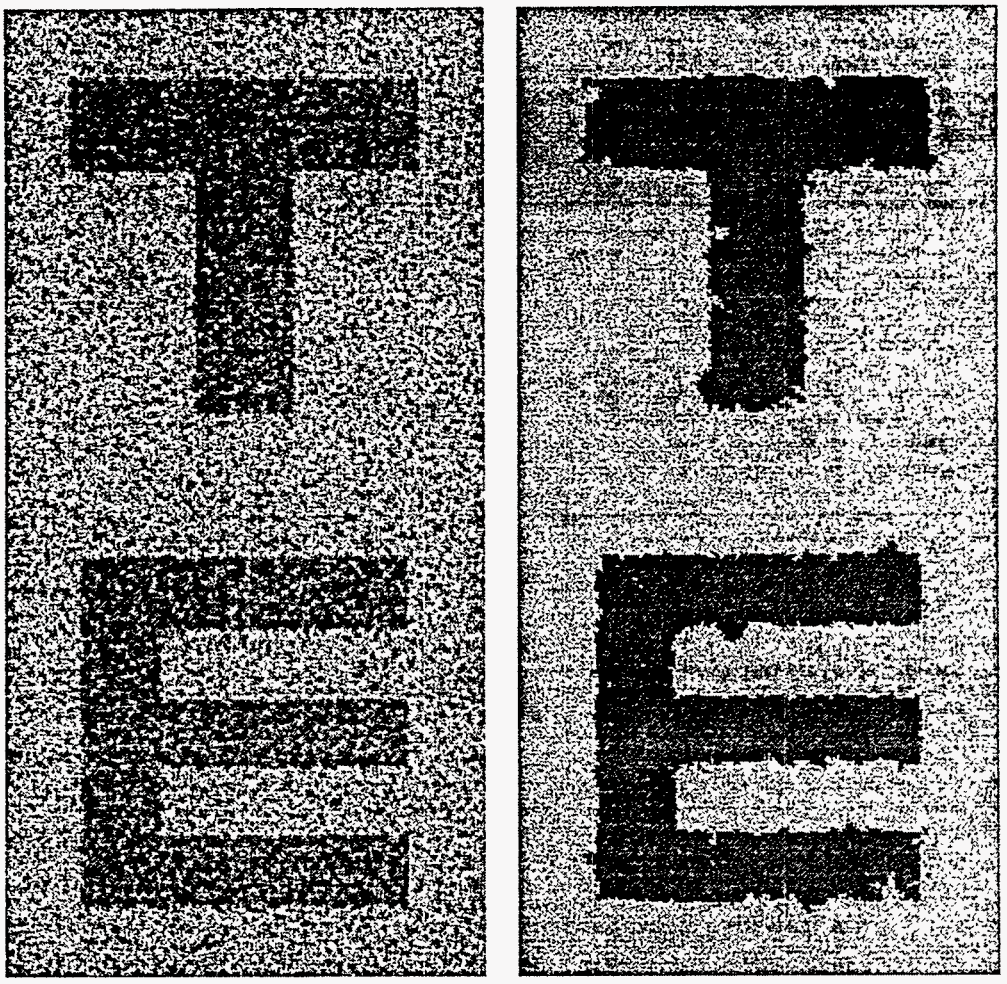


Figure 6: The averaged noisy image and segmentation results for $\sigma = 50$.

INTERNAL DISTRIBUTION

1. B. R. Appleton
2. J. Barhen
3. M. Beckerman
4. J. R. Einstein
5. C. W. Glover
6. X. Guan
7. J. P. Jones
8-10. R. C. Mann
11. S. Matis
12. S. A. McKenney
13. R. J. Mural
14. E. M. Oblow
15. C. E. Oliver
16. S. Petrov
17. N.S.V. Rao
18. D. B. Reister
19. M. B. Shah
20. R. F. Sincovec
21-22. E. C. Uberbacher
23-27. X. Ying
29. EPMD Report Office
30-31. Laboratory Records Department
32. Laboratory Records, ORNL-RC
33. Document Reference Section
34. Central Research Library
35. ORNL Patent Office

EXTERNAL DISTRIBUTION

36. Office of Assistant Manager for Energy Research and Development, Oak Ridge Operations, U.S. Department of Energy, P.O. Box 2008, Oak Ridge, TN 37831
37. Martha A. Krebs, Director, Office of Energy Research, FORS, ER-1, Dept. of Energy, Washington, DC 20545
38. Mr. Joseph L. Mundy, Senior Research Fellow, General Electric Corporate Research and Development Center, 1 River Road, Schenectady, NY 12309
39. Dr. Rama Chellappa, Center for Automation Research, University of Maryland, College Park, MD 20742
40. John Wooley, Acting Director, Office of Energy Research, FORS, ER-1, U.S. Department of Energy, Washington, DC 20545
41. Mr. William Glatz, NPIC, Room 3N500, National Exploitation Laboratory, P.O. Box 70967 Southwest Station, Washington, DC 20024-0967
42. M. D. Zorn, Lawrence Berkeley Laboratory, MS 50B-3216, 1 Cyclotron Road, Berkeley, CA 94720
43. Dr. Thomas Strat, Software & Intelligent Systems Technology Office, Advanced Research Projects Agency, 3709 North Fairfax Drive, Arlington, VA 22203
44. Mr. Donald Gerson, Office of Research and Development, P.O. Box 4132, Washington, DC 20505
45. Dr. Kevin Boywer, University of South Florida, 4202 East Fowler Avenue, Tampa, FL 33620-5399
46. Dr. Sudeep Sarkar, Department of Computer Science and Engineering, University of South Florida, 4202 East Fowler Avenue, ENB118, Tampa, FL 33620-5399
47. Mr. John Blair, TBX Technologies, 25 Moore Road, Wayland, MA 01778
48. Dr. Avi Kak, Department of Electrical Engineering, Purdue University, Lafayette, IN 47907
49. Dr. Peter Allen, Department of Computer Science, Columbia University, 450 Computer Science Bldg., New York, NY 10027

50. Dr. J. K. Aggarwal, Computer and Vision Research Center, University of Texas, Austin, TX 78712-1084
51. Office of Scientific and Technical Information, P.O. Box 62, Oak Ridge, TN 37831
52. Dr. O. P. Manley, Division of Engineering and Geosciences, ER-15, Department of Energy, Washington, DC 20874-1290