

Boundary and Object Detection in Real World Images

YORAM YAKIMOVSKY

Jet Propulsion Laboratory, Pasadena, California

ABSTRACT A computer solution to the problem of automatic location of objects in digital pictures is presented. A self-scaling local edge detector that can be applied in parallel on a picture is described. Clustering algorithms and sequential boundary following algorithms process the edge data to local images of objects and generate a data structure that represents the imaged objects.

KEY WORDS AND PHRASES scene analysis, image processing, clustering, statistical decision analysis, maximum likelihood test

CR CATEGORIES 3 63, 3 83

1. Introduction

A substantial amount of research has been done in developing techniques for locating objects of interest automatically in digitized pictures. Drawing the boundaries around objects is essential for pattern recognition, object tracking, image enhancement, data reduction, and various other applications. References [18-20] constitute a good survey of research and applications in image processing and picture analysis.

Most researchers of picture analysis have assumed that (1) the image of an object is more or less uniform or smooth in its local properties (that is, illumination, color, and local texture are smoothly changing inside the image of an object); and (2) there is detectable discontinuity in local properties between images of two different objects. We will adopt these two assumptions in this paper and assume no textural image (see [1] for an example of texture image analysis that does not make these assumptions).

The work on automatic location of objects in digitized images has split into two approaches: edge detection and edge following versus region growing. Edge detection applies local independent operators over the picture to detect edges and then uses algorithms to trace the boundaries by following the local edge detected. A recent survey of literature in this area is given in [7]. The region growing approach uses various clustering algorithms to grow regions of almost uniform local properties in the image (see [5, 2, 11, 24] for typical applications). More detailed references will be given later.

In this paper the two approaches are combined to complement each other; the result is a more powerful mechanism to segment pictures into objects. We developed a new edge detector and combined it with new region growing techniques to locate objects; in so doing we resolved the confusion in regular edge following that results where more than one isolated object on a uniform background is in the scene (see [17]).

This report describes the following contributions: (1) a new and "optimal" (given certain assumptions) edge detector; (2) a simple one-pass region growing algorithm that

Copyright © 1976, Association for Computing Machinery, Inc. General permission to republish, but not for profit, all or part of this material is granted provided that ACM's copyright notice is given and that reference is made to the publication, to its date of issue, and to the fact that reprinting privileges were granted by permission of the Association for Computing Machinery.

This paper presents the results of one phase of research carried out at the Jet Propulsion Laboratory, California Institute of Technology, under Contract NAS7-100, sponsored by the National Aeronautics and Space Administration.

Author's present address: Ph D-M D Program, University of Miami School of Medicine, Miami, FL 33152

is implemented on a minicomputer, utilizing the edge detector output; (3) the application of path generator algorithms and "shortest path" algorithms for boundary following to close open-edge lines into boundaries around regions; (4) special purpose region growing intended to close open edges (cracks); and (5) a special clustering algorithm that simplifies the region structure resulting from the application of (1) through (4)

2. Definition of Terms

The input is expected to be in the matrix form $V(i,j)$, $i = 1, \dots, N$, $j = 1, \dots, M$, where V is a vector in R^n and n is a function of the sensory system, usually 1 (gray level picture), 3 (color or x, y, z coordinates of the surface in the scanning direction), or 6 (color and 3-D information). An edge unit separates two adjacent matrix points; that is, an edge unit is between (i,j) and $(i + 1,j)$ or between (i,j) and $(i,j + 1)$ for some i,j (see Figure 1).

An edge unit is usually adjacent on both ends to other edge units. There are 64 combinations of edge units continuing an edge unit, since each of the edge units $e_1, e_2, e_3, e_1', e_2', e_3'$ in Figure 1 may or may not exist.

Two points on the grid (I, J) and (K, L) are said to be in the same region if there is a path sequence $(i_1, j_1), \dots, (i_n, j_n)$ such that $i_1 = I, j_1 = J, i_n = K$, and $j_n = L$, where (i_m, j_m) is adjacent to (i_{m+1}, j_{m+1}) for $m = 1, \dots, n - 1$ and there is no edge unit between the two. A region will be a maximum set of points satisfying that property.

An edge line (or an edge) between region R_1 and region R_2 is the maximal sequence of adjacent edge units such that each edge unit in the sequence is between two matrix points, one belonging to R_1 and the other belonging to R_2 . It is possible that an edge line is inside a region ($R_1 = R_2$).

An edge line that is between two different regions is called a boundary. An edge line that is inside a region is called a crack. An open crack is a crack in which at least one end terminates without connecting to any edge line. A closed crack is one in which each end terminates on another edge line. For instance cracks appear when an object is smoothly disappearing into the background on one side and has detectable discontinuity on the other side, as shown in Figure 2.

Using the above definitions, this report first presents an edge detector that detects edge units in parallel locally on the whole image. Then a region grower that groups matrix points into regions and edge units into boundaries and cracks is presented. A local region grower that tries to break a region with a crack in it into two regions for which the crack is part of the common boundary is presented next. Alternatively, an open crack extending algorithm is suggested to connect the open edge unit of the crack to another edge line.

3. The Local Edge Detector

The edge operator is a detector of local discontinuity in an image. When applied between two adjacent points such as (i, j) and $(i + 1, j)$, it should return a value that will measure the confidence that there is an edge between (i, j) and $(i + 1, j)$. Since we work with noisy input to achieve reliability, the operator must look at two 2-dimensional (2-D) neighborhoods N_1 and N_2 to obtain a reliable value. Neighborhood N_1 includes (i, j) and a few adjacent points; N_2 includes $(i + 1, j)$ and a few adjacent points; and $N_1 \cap N_2 = 0$. As a

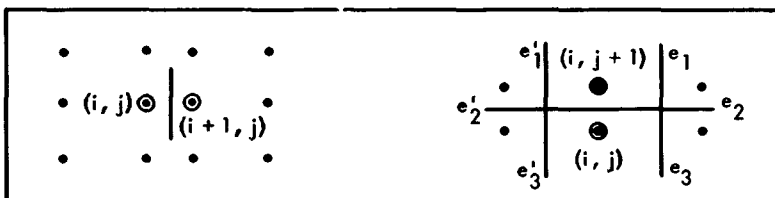


FIG 1 Edge unit structure

result the value returned will measure the confidence that the neighborhoods belong to images of different objects

Edge detection is actually composed of three components: (1) measuring differences between image structures in the two neighborhoods, (2) selecting the proper neighborhoods; and (3) locking on the exact position of the edge. Discussion of each of these steps follows.

4. Measuring Differences in Structure Between Two Neighborhoods

Any techniques that measure structural differences must make some assumptions (explicitly or implicitly) concerning the structure of an edge and the area inside a region. Binford and Hershkovitz [4] suggest three possible ideal edges defined by the intensity profile on a normal-to-the-edge line (Figure 3).

All of these idealized edges are in reality washed with Gaussian noise on both sides, where the noise is both hardware noise and the result of surface irregularities. Basically, the decision is between two hypotheses:

H_0 : the readings in N_1 and N_2 are taken from the same object;

H_1 : the readings in N_1 and N_2 are taken from different objects

Neighborhoods N_1 and N_2 are the neighborhoods mentioned in Section 3, and the decision as to how to choose them will be described in Section 5.

An optimal (best for its size) decision between H_0 and H_1 will utilize the maximum

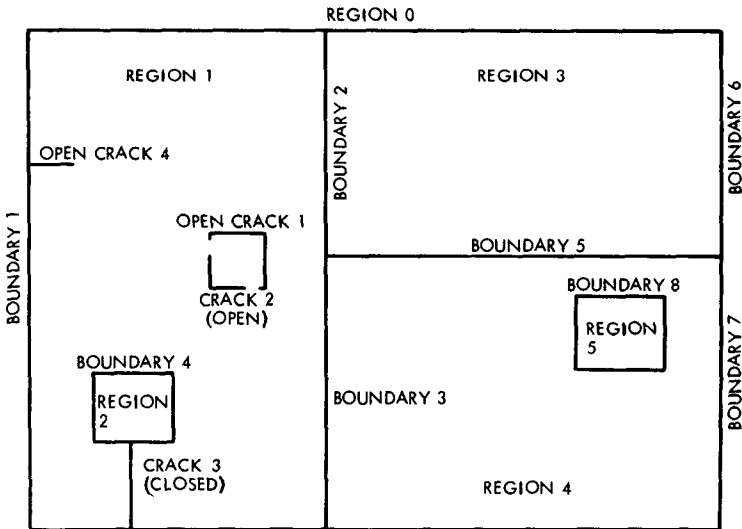


FIG 2 Illustration of terms

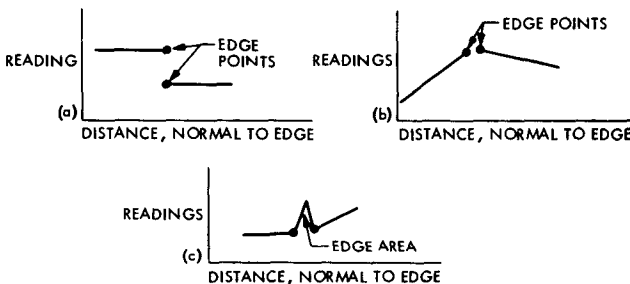


FIG 3 Typical edges (a) idealized step edge (dominant edge type in visual images), (b) pure gradient edge (corners are especially frequent in analysis of 3-D images when direct measure of distance is available), (c) spike edge (appears frequently in corner edges in visual images)

likelihood ratio as follows: Let P_0 be the maximum likelihood estimate of the structure (reading in N_1 and N_2), given that H_0 is true, and let P_1 be the maximum likelihood estimate of the structure, assuming that H_1 is true. Then choose H_1 when $P_1/P_0 > K$, choose H_0 when $P_1/P_0 < K$, and choose at random when $P_1/P_0 = K$.

This decision will be optimal for a given allowed probability of false negative (see the Neyman-Pearson Test [8, p. 55]); hence if the structure assumptions are valid we have an ideal edge detector, given only readings in N_1 and N_2 . (We deal with Gaussian probabilities; hence we ignore $P_1/P_0 = K$.) The conclusion is that P_1/P_0 is the best measure of the edge strength. Following are two examples of applying these principles to the edges of types (a) and (b) in Figure 3.

Example 1 Assume that the edges and surfaces will be of type (a) as in Figure 3 with added white noise which is object-dependent. Then H_0 and H_1 become

H_0 : the readings in both N_1 and N_2 are independently taken from the same normal distribution $N(\mu_0, \sigma_0)$ with unknown μ_0, σ_0 ;

H_1 : the readings on N_1 are independently taken from normal distribution $N(\mu_1, \sigma_1)$, the readings on N_2 are taken from normal distribution $N(\mu_2, \sigma_2)$, and (μ_1, σ_1) need not be equal to (μ_2, σ_2) .

To apply the maximum likelihood ratio principle we must find a maximum likelihood estimate for (μ_0, σ_0) , (μ_1, σ_1) , and (μ_2, σ_2) . Given (x_1, \dots, x_n) readings taken from a normal distribution with unknown (μ, σ) , the maximum likelihood estimates for (μ, σ) are $(\bar{\mu}, \bar{\sigma})$. When

$$\bar{\mu} = \frac{1}{n} \sum_{i=1}^n x_i \quad \text{and} \quad \bar{\sigma}^2 = \frac{1}{n} \sum_{i=1}^n (x_i - \bar{\mu})^2,$$

the probability density that a reading x_i is generated by $N(\bar{\mu}, \bar{\sigma})$ is

$$(1/\sqrt{(2\pi)\sigma}) \cdot e^{-(x_i - \bar{\mu})^2/2\bar{\sigma}^2}$$

Assuming that (x_1, \dots, x_n) are independently taken from normal distribution $N(\bar{\mu}, \bar{\sigma})$, then the joint probability density of generating the combined reading is the product of the individual terms.

$$\begin{aligned} P_{\max} &= P_{(\bar{\mu}, \bar{\sigma})}(x_1, \dots, x_n) \\ &= [1/(\sqrt{(2\pi)\bar{\sigma}})^n] e^{-(1/2\bar{\sigma}^2) \sum_{i=1}^n (x_i - \bar{\mu})^2} \\ &= [1/\sqrt{(2\pi) \cdot \sigma^n}] e^{-n\sigma^2/2\sigma^2} \\ &= (1/2\pi^n) \cdot e^{n/2} \cdot (1/\sigma^n). \end{aligned}$$

Hence if the readings on N_1 are (x_1, \dots, x_m) and on N_2 (y_1, \dots, y_n) , then on N_1 ,

$$\mu_1 = \frac{1}{m} \sum_{i=1}^m x_i, \quad \sigma_1^2 = \frac{1}{m} \sum_{i=1}^m (x_i - \mu_1)^2, \quad P_1 = (1/(2\pi)^{m/2}) \cdot e^{-m/2} \cdot 1/\sigma_1^m;$$

on N_2 ,

$$\mu_2 = \frac{1}{n} \sum_{i=1}^n y_i, \quad \sigma_2^2 = \frac{1}{n} \sum_{i=1}^n (y_i - \mu_2)^2, \quad P_2 = (1/(2\pi)^{n/2}) \cdot e^{-n/2} \cdot 1/\sigma_2^n;$$

and on N_1 combined with N_2 ,

$$\begin{aligned} \mu_0 &= (m\mu_1 + n\mu_2)/(m + n), \\ \sigma_0^2 &= [m\sigma_1^2 + n \cdot \sigma_2^2 + m(\mu_0 - \mu_1)^2 + n(\mu_0 - \mu_2)^2]/(m + n), \\ P_0 &= \frac{1}{(2\pi)^{(m+n)/2}} \cdot e^{-(m+n)/2} \cdot 1/\sigma_0^{m+n}. \end{aligned}$$

Where H_1 holds we further assume that the readings on the two neighborhoods are independent; this results in the joint probability density of P_1 and P_2 being the product of $P_1 \cdot P_2$. Hence the maximum likelihood ratio is $P_1 \cdot P_2 / P_0$. Squaring this expression, which saves computations of square roots, results in the following expression for the edge value:

$$P_1^2 \cdot P_2^2 / P_0^2 = (\sigma_0^2)^{m+n} / (\sigma_1^2)^m \cdot (\sigma_2^2)^n.$$

Note that the edge value suggested is self-scaling with respect to noise and texture: In areas where $\sigma_1 \cong \sigma_2 \cong \sigma_0 \gg 0$ (highly textured areas or the result of noisy hardware) the edge value will be low, near 1, while any small steps in almost uniform areas will be recognized early. In practice, we computed the variance of noise in the hardware by sampling over time the same points in static scenes. The computed variance is taken always to be at least the hardware noise. Thus divisions by zero in pathological cases were prevented.

At this point it may be worthwhile to compare our approach with that of [9]. Both try to use a maximum likelihood ratio to compute scores for an edge. But while we have a simple model and a practical way of computing the confidence, [9] assumes a priori deterministic classification of all possible idealized noise free structures into edges and no edges. Then, for a given reading structure the noise assumption is used to compute the probability of all idealized structures that could have caused the readings. These probabilities are used to decide whether or not the readings represent an edge.

It should be mentioned that other statistical techniques, e.g. [4, 22], were used for edge detection, but none of the edge detectors that appeared in the literature used the maximum likelihood test for edge value.

Example 2 Here we assume that each matrix point $\mathbf{V}(i, j)$ is a 3-D vector (x, y, z) . Actually the raw readings are just distance $R(i, j)$, but to avoid a strong dependency on the sensor position, $R(i, j)$ is transformed into (x, y, z) . This is the form of input read from such a device as radar, which measures distances to surfaces, or from devices that measure the time of flight of light (laser) beams to an object. The i, j corresponds to vertical and horizontal steps in the scanning angle. In that model two adjacent neighborhoods on the matrix N_1 and N_2 have readings $(x_1, y_1, z_1), \dots, (x_n, y_n, z_n)$ in N_1 and $(x_1^1, y_1^1, z_1^1), \dots, (x_m^1, y_m^1, z_m^1)$ in N_2 . We assume that objects are almost planar locally with added white noise with mean 0 to position readings. That is, if we read $(x_1, y_1, z_1), \dots, (x_n, y_n, z_n)$ in a small neighborhood on an object we have a, b, c, d, σ such that $a^2 + b^2 + c^2 = 1$ and $ax_i + by_i + cz_i + d + N(0, \sigma) = 0, i = 1, \dots, n$.

With this assumption the edge detection decision will be a choice between H_0 and H_1 :

H_0 : The readings in the two neighborhoods are taken from the same plane. That is, the readings on both N_1 and N_2 satisfy for some $(a_0, b_0, c_0, d_0, \sigma_0)$ the equation $a_0x + b_0y + c_0z + d_0 + N(0, \sigma_0) = 0$, where $a_0^2 + b_0^2 + c_0^2 = 1$ for all (x, y, z) readings in N_1 and N_2 .

H_1 : There are two not necessarily equal planar fits for the readings on N_1 and on N_2 . That is, there are $(a_1, b_1, c_1, d_1, \sigma_1)$ for N_1 and $(a_2, b_2, c_2, d_2, \sigma_2)$ for N_2 such that $a_1^2 + b_1^2 + c_1^2 = 1; a_2^2 + b_2^2 + c_2^2 = 1; a_1x_i + b_1y_i + c_1z_i + d_1 + N(0, \sigma_1) = 0, i = 1, \dots, n; a_2x_i^1 + b_2y_i^1 + c_2z_i^1 + d_2 + N(0, \sigma_2) = 0, i = 1, \dots, m$

To apply the Neyman-Pearson principle for this case we want to find maximum likelihood estimates. Maximum likelihood estimates a_1, b_1, c_1, d_1 will be

$$V_1 = \sum_{i=1}^n (a_1x_i + b_1y_i + c_1z_i + d_1)^2$$

$$= \min_{\substack{a,b,c,d \\ a^2+b^2+c^2=1}} \sum_{i=1}^n (ax_i + by_i + cz_i + d)^2 \quad \text{and} \quad \sigma^2 = V_1/n.$$

Solving for the optimal (a_1, b_1, c_1, d_1) is a relatively straightforward process. Once they are found, the maximum likelihood estimate for N_1 is

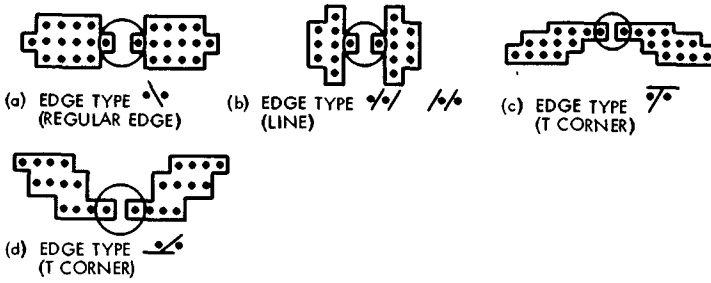


FIG 4 Typical neighborhoods for edge detection

$$P_1 = 1/\sqrt{(2\pi^n) \cdot \sigma_1^n \cdot e^{-n/2}}$$

Hence we have the expression that tests for an edge the following way: If

$$V_0^{m+n}/(V_1^n V_2^m) \geq K^2,$$

decide for H_1 ; otherwise decide for H_0 .

Note that (x, y, z) may be replaced by (t, j, g) in regular black and white pictures, in which case we have a regular picture edge operation that can handle edges of type (b) in Figure 3. Somewhat similar applications have been reported [21 and 4] for detection of gradient edges (Figure 3(b)) This edge operator has not yet been incorporated in our system.

5. Neighborhood Selection

In the previous discussion on decision criteria, we deliberately omitted the question of how to choose the test neighborhoods. This is another variant of the properties that we want the edges to have. The edge value for a vertical edge between two horizontally adjacent points is taken to be the strongest case for an edge computed on the four pairs of neighborhoods (a)-(d) in Figure 4. Taking the maximum of the maximum likelihood ratio estimate for an edge among the four values computed for the four neighborhoods is similar to the approach advocated in [6].

A completely symmetric configuration is used to measure the confidence value of a horizontal edge unit between two vertically adjacent points. The choice of neighbors is of an experimental nature, and it worked for our problems. Other problem-dependent neighborhood choices are possible, and they will work for the specific edge structure in mind, as shown by the examples in Figure 5. In choosing the size of a neighborhood a reasonable balance between noise and size of object should be achieved. The bigger the neighborhoods the less sensitive to noise the decision will be, but the small objects may be lost.

At this point it is worthwhile to refer to the edge detector developed by Hueckel [13]. He found an elegant technique that can be used to compute the best fitting 2-D step function.

$$STEP_{a,b,c,d,e}(t, j) = \begin{cases} d & \text{if } at + bj \geq c \\ e & \text{if } at + bj < c \end{cases}$$

is defined over a disk

$$DISK(i_0, j_0, \gamma) \triangleq \{ (i, j) \mid (i - i_0)^2 + (j - j_0)^2 \leq \gamma \},$$

which corresponds to an edge line $at + bj = c$ where the brightness on one side is d and on the other e . The step function is selected in an attempt to minimize the expression for a given signal $g(i, j)$ in the disk,

$$\sum_{(i,j) \in DISK} (g(i, j) - STEP_{a,b,c,d,e}(i, j))^2,$$

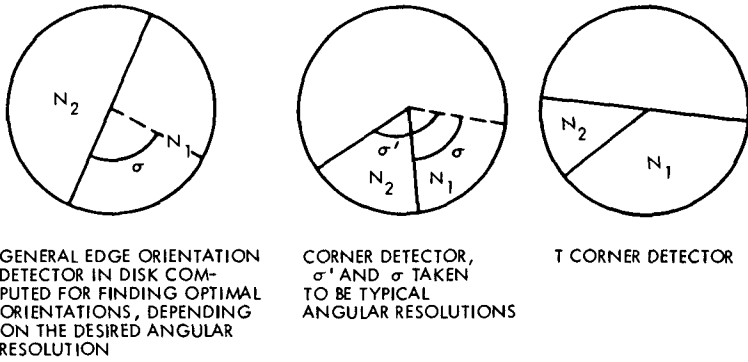


FIG 5 Extended neighborhoods set

over all possible step functions. He took the parameters of a, b, c, d, e to be the parameters of the "best possible" edge passing through the disk. The edge value was then defined as $|d - e| / DIS$, which is a different measure than ours. An efficient method of finding good approximation of those parameters was developed. Since our measure of edge strength is more complicated, it is unlikely that an elegant and simple way of finding an optimal edge through a disk using our measure of edge strength is achievable. However, given a suggested edge structure, our approach can be used immediately to provide a model driven confidence evaluation in the existence of the suggested edge. For the suggested (a, b, c, d, e) edge parameter, let

$$\begin{aligned}
 N_2 &= \sum_{\substack{a+b \geq c \\ (i,j) \in DISK}} 1, & \mu_2 &= d, & \sigma_2^2 &= \sum_{\substack{a+b \geq c \\ (i,j) \in DISK}} (g(i, j) - d)^2 / N_2; \\
 N_1 &= \sum_{\substack{a+b < c \\ (i,j) \in DISK}} 1, & \mu_1 &= e, & \sigma_1^2 &= \sum_{\substack{a+b < c \\ (i,j) \in DISK}} (g(i, j) - e)^2 / N_1; \\
 N_0 &= N_1 + N_2, & & & \sigma_0^2 &= \sum_{(i,j) \in DISK} (g(i, j) - \mu_0)^2 / N_0. \\
 \mu_0 &= (N_2 \cdot \mu_2 + N_1 \cdot \mu_1) / N_1 + N_2, & & & &
 \end{aligned}$$

Then

$$STRENGTH = \left(\sigma_0^2\right)^{\frac{1}{2}} / \left(\sigma_1^2\right)^{\frac{1}{2}} \left(\sigma_2^2\right)^{\frac{1}{2}},$$

which uses again the theoretical superior maximum likelihood test.

6. Locking on a Detected Edge

The computed edge value is usually not sufficient to determine the location of the edges. The values that are computed usually look like those in Figure 6

One way of forcing the edge to be well defined is to constrain it to be a local maximum in addition to having a confidence value higher than a certain threshold. This is, of course, extremely important for locking on the center of the edge (see [14, p. 382]). Usually there is still some local ambiguity on the location of the edge, and for many practical reasons it is better to treat the area around an edge as ambiguous. The source of the problems here is that because of computing time constraints it is impossible to find a global optimum for edge lines using all available data, and it is necessary to use only local information for evaluating the edge units at this level. In our system the decision concerning the exact location of the edge was left for the region grower described in Section 7. Figure 7 illustrates the possible 2-D ambiguity.

The search for a maximum may be used for special purpose edge detection. For instance, if we look only for one dark stripe crossing a white background, forcing the

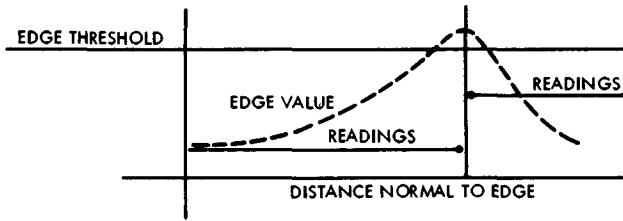


FIG 6 An ideal edge value cross section

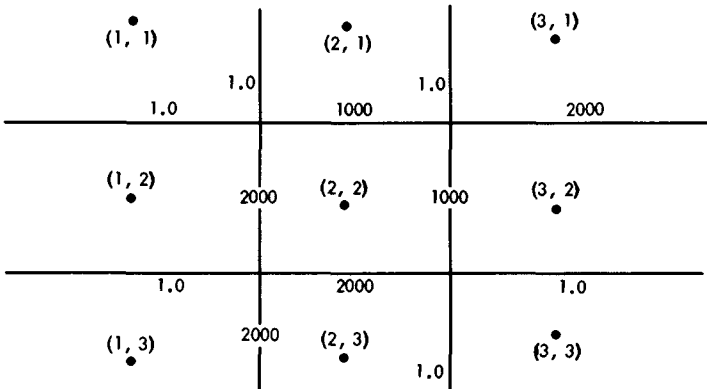


FIG 7 Region growing ambiguity example. The (i, j) are point numbers and the values are edge unit values. Clearly points (1, 1) (1, 2) (1, 3) (2, 1) (3, 1) should be in one region and (3, 2) (3, 3) (2, 3) in another, but where (2, 2) should be is totally ambiguous (assuming that single point regions are not allowed)

edge to be the absolute maximum or minimum on a horizontal line in the image (keeping track of the direction of the change) will supply the appropriate pair of edges on each of the horizontal lines.

7 Region Growing

The application of an edge detector results in two new matrices in addition to the matrix $V(i, j)$ of raw data. The first is $EV(i, j)$, which is the measure of the confidence that there is an edge unit between (i, j) and $(i, j + 1)$; the second is $EH(i, j)$, which measures the confidence that there is an edge unit between (i, j) and $(i + 1, j)$. $EV(i, j)$ and $EH(i, j)$ may include extra bits as determined by the direction of the change on that suggested edge unit.

This output as it stands is not sufficient for application of pattern recognition and various picture quantitative analysis tasks. Outlines of objects are needed to recognize features. One way of achieving these is to use a region grower that will outline objects by clustering points into regions. This approach has been used for picture analysis [5, 2, 11, 24]. The basic conclusion of these works is that without using semantic information, which is the knowledge of the subject of the picture, clustering cannot create perfect outlines. More recent work [8, p. 324, and 12] has introduced new techniques of clustering that provide more flexibility and may upgrade clustering performance for images.

Here we introduce a new algorithm for clustering based on a search for "valleys" of edge values in a picture. If random access is allowed, a relatively simple algorithm that starts and climbs from local minima of edge values can be implemented. Because of a lack of storage capacity on our minicomputer, and in an attempt to use data as it is sequentially digitized from the video signal, a one-pass algorithm to generate regions corresponding to valleys was implemented. This is the first time to our knowledge that this approach has been used.

Most works on region growing (ours included) lack the capacity to make use of the shape of the growing object. An alternative approach to region growing is "edge following" [17, 15, 10, and 3]. The basic idea in edge following is to detect a discontinuity, trace it, and in this way define edge lines. Unfortunately the work in edge following lacks an effective way of tying region shape properties into their decision processes and output.

Let us start by describing a one-pass algorithm that transforms the edge into data structures of regions, boundaries, closed cracks, and open cracks, creating as by-products two arrays, $FH(i, j)$ and $FV(i, j)$, where $FH(i, j)$ means that the program puts an edge unit between $(i - 1, j)$ and (i, j) and $FV(i, j)$ means an edge unit between (i, j) and $(i, j - 1)$.

To facilitate the description of the decision mechanism for placing edges, we need to define a few new terms. Let $T > 0$ be the edge confidence threshold; then:

(1) d is the distance between two adjacent grid points:

$$d((i, j), (i - 1, j)) \triangleq d((i - 1, j), (i, j)) \triangleq (\text{if } EH(i, j) \leq T \text{ then } 0 \text{ else } EH(i, j)).$$

$$d((i, j), (i, j - 1)) \triangleq d((i, j - 1), (i, j)) \triangleq (\text{if } EV(i, j) \leq T \text{ then } 0 \text{ else } EV(i, j)).$$

(2) $Reg(i, j)$ is the region to which the point (i, j) belongs. ($Reg(i, j)$ is not defined to all points until the program is finished.)

$$(3) Val(i, j) = \underset{|i-k|+|j-m|=1}{\text{Min}} d((i, j), (k, m)).$$

There is no edge unit between (i, j) and (k, m) . This value will be $+\infty$ if (i, j) is the only point in its region.

$$(4) Val(Reg_1) = \underset{(i,j)}{\text{Min}} (Val(i, j))$$

(5) A point P will be the minimum point for its region if $Val(P) = Val(Reg(P))$.

The algorithm is designed so that at each state there is always a nondecreasing edge distance value path from each minimum of any region to any other point in the region and the path enters that point from its minimum direction.

That is, if P and Q are two points such that $Reg(P) = Reg(Q)$ and $Val(P) = Val(Reg(P))$, then there is a path (x_1, x_2, \dots, x_n) such that

- (a) $x_1 = P, x_n = Q$;
- (b) $Reg(x_i) = Reg(P), i = 1, \dots, n$;
- (c) x_i adjacent to $x_{i+1}, d(x_{i+2}, x_{i+1}) \geq d(x_{i+1}, x_i)$;
- (d) $d(x_n, x_{n-1}) = Val(x_n)$.

We say that if such a path exists, Q is *reachable* from P . That is, two points are in the same region if you can get from one to the other in a path that does not cross a ridge of edge values.

8. Algorithm Description

The program scans the image line by line from left to right. The scanning is such that when point (i, j) is processed, the program has already worked on all points (i_1, j_1) such that $(j_1 < j)$ or $(j = j_1 \text{ and } i_1 < i)$. At each point one of the conditions in Figure 9 exists, and the algorithm treats them as described in Figure 9 to grow the regions, boundaries, and crack lines.

Assume the program is processing point (i, j) .

Let D_1 be a Boolean variable set to **true** if the program is not going to put an edge unit between (i, j) and $(i, j - 1)$ and set to **false** otherwise, and let D_2 be a Boolean variable set to **true** if the program is not going to put an edge unit between (i, j) and $(i - 1, j)$ and set to **false** otherwise. Let $R_1 = Reg(i, j - 1)$ and $R_2 = Reg(i - 1, j)$ (see Figure 8).

The decision on the values of D_1 and D_2 is described by the following Algol-like program:

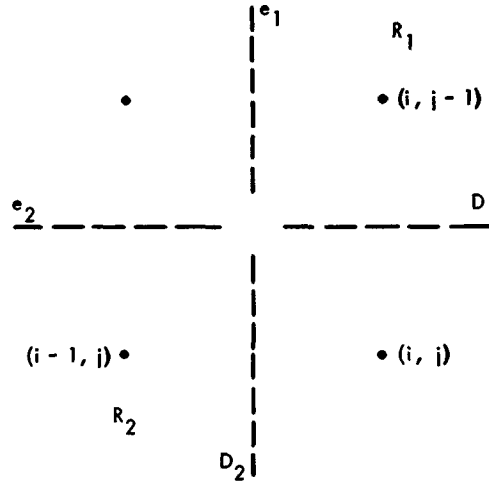


FIG 8 Algorithm terms definition

begin

Boolean $good_down_1, bad_down_1, up_1, good_down_2, bad_down_2, up_2,$

$good_down_1 \leftarrow (d((i, j), (i, j - 1)) \leq Val(i, j - 1) \wedge ((Val(i, j - 1) \leq Val(R_1))),$

Comment $good_down_1$ is **true** if point (i, j) is going to become a new minimum for R_1 (the region above) and it is adjacent to an old minimum, hence any point of R_1 reachable from the old adjacent minimum will be reachable from the new,

$bad_down_1 \leftarrow (d((i, j), (i, j - 1)) < Val(i, j - 1)) \wedge ((Val(i, j - 1) > Val(R_1))),$

Comment This variable is **true** if (i, j) is not reachable from all minima of R_1 going through $(i, j - 1)$,

$up_1 \leftarrow d((i, j), (i, j - 1)) \geq Val(i, j - 1),$

Comment This variable is **true** if point (i, j) is reachable from any minimum of R_1 by continuing the path that leads from that minimum to $(i, j - 1)$,

$good_down_2 \leftarrow (d((i, j), (i - 1, j)) \leq Val(i - 1, j)) \wedge ((Val(i - 1, j) \leq Val(R_2)));$

Comment This variable is **true** if point (i, j) is going to be a new minimum for R_2 (the region minimum, to the side) and is adjacent to an old minimum of R_2 , hence any point reachable from the adjacent old minimum will be reachable from (i, j) ,

$bad_down_2 \leftarrow (d((i, j), (i - 1, j)) < Val(i - 1, j)) \wedge (Val(i - 1, j) > Val(R_2));$

Comment This variable is **true** if (i, j) is not reachable from all minima of R_2 through $(i - 1, j)$,

$up_2 \leftarrow d((i, j), (i - 1, j)) \geq Val(i - 1, j),$

Comment This variable is **true** if point (i, j) is reachable from any minima of R_2 by continuing the path that leads from that minimum to $(i - 1, j)$,

If $good_down_1 \wedge good_down_2$ **then** $D_1 \leftarrow D_2 \leftarrow$ **true**

else

If $good_down_1 \wedge bad_down_2$ **then** $D_1 \leftarrow D_2 \leftarrow$ **true**

else

If $good_down_1 \wedge bad_down_2$ **then begin** $D_1 \leftarrow$ **true,**

$D_2 \leftarrow$ **false, end else if** $good_down_1 \wedge up_2$ **then**

begin

if $d((i, j), (i, j - 1)) \geq d((i, j), (i - 1, j))$

then $D_1 \leftarrow D_2 \leftarrow$ **true**

else begin $D_1 \leftarrow$ **true, $D_2 \leftarrow$ **false, end,****

end

else if bad_down_1 **then begin if** $good_down_2 \vee up_2$ **then begin**

$D_1 \leftarrow$ **false,**

$D_2 \leftarrow$ **true,**

end

else begin

$D_1 \leftarrow$ **false,**

$D_2 \leftarrow$ **false, end;**

end

else if $up_1 \wedge good_down_1$ **then begin**

if $d((i, j), (i - 1, j)) \geq d((i, j), (i, j - 1))$

then $D_1 \leftarrow D_2 \leftarrow$ **true**

else begin $D_2 \leftarrow \text{true}$, $D_1 \leftarrow \text{false}$, end,
end

else if $up_1 \wedge up_2$ then, if $R_1 = R_2$ then $D_1 \leftarrow D_2 \leftarrow \text{true}$ else

if $d((t, j), (t - 1, j)) \geq d((t, j), (t, j - 1))$

then begin $D_1 \leftarrow \text{true}$, $D_2 \leftarrow \text{false}$, end

else begin $D_1 \leftarrow \text{false}$; $D_2 \leftarrow \text{true}$; end

end

Comment In that case only one of D_1 and D_2 can be true, otherwise we cannot guarantee entrance through a minimum value from all minima of both R_1 and R_2 ,

else if $up_1 \wedge \text{bad-down}_2$ then begin $D_1 \leftarrow \text{true}$, $D_2 \leftarrow \text{false}$; end

$Val(t, j) \leftarrow \infty$;

if D_1 then begin

$Val(t, j - 1) \leftarrow \text{Min}(d((t, j), (t, j - 1))), Val(t, j - 1))$;

$Val(t, j) \leftarrow d((t, j), (t, j - 1))$,

$Val(R_1) \leftarrow \text{Min}(Val(R_1), Val(t, j))$,

end,

if D_2 then begin

$Val(t - 1, j) \leftarrow \text{Min}(d((t, j), (t - 1, j)), Val(t - 1, j))$,

$Val(t, j) \leftarrow \text{Min}(Val(t, j), d((t, j), (t - 1, j)))$,

$Val(R_2) \leftarrow \text{Min}(Val(R_2), Val(t, j))$;

end,

if not $(D_1 \vee D_2)$ then $Val(\text{Reg}(t, j)) \leftarrow \infty$,

The e_1 and e_2 (see Figure 8) may exist or not, and as a result there are four starting conditions. The program may put D_1 , D_2 , D_1 , and D_2 or none of them, and hence there are 16 cases in a point. (See Figure 9 for a brief description of the different cases.)

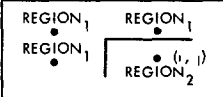
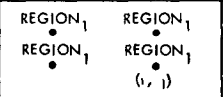
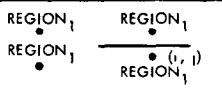
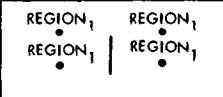
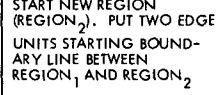
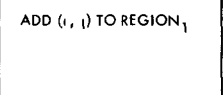
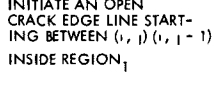
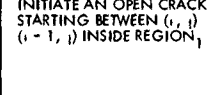
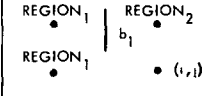
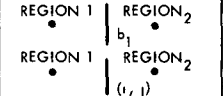
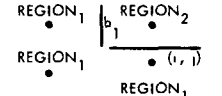
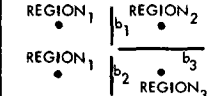
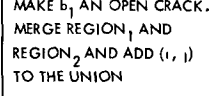
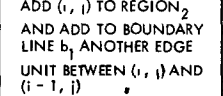
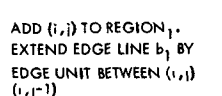
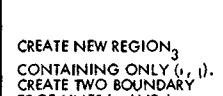
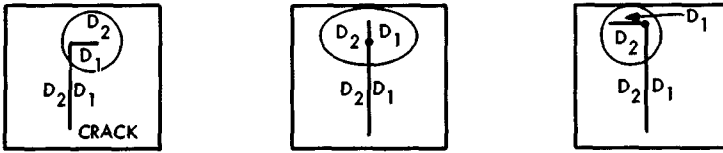
 <p>START NEW REGION (REGION₂). PUT TWO EDGE UNITS STARTING BOUNDARY LINE BETWEEN REGION₁ AND REGION₂</p>	 <p>ADD (i, j) TO REGION₁</p>	 <p>INITIATE AN OPEN CRACK EDGE LINE STARTING BETWEEN (i, j) (i, j - 1) INSIDE REGION₁</p>	 <p>INITIATE AN OPEN CRACK STARTING BETWEEN (i, j) (i - 1, j) INSIDE REGION₁</p>
 <p>MAKE b₁ AN OPEN CRACK. MERGE REGION₁ AND REGION₂ AND ADD (i, j) TO THE UNION</p>	 <p>ADD (i, j) TO REGION₂ AND ADD TO BOUNDARY LINE b₁ ANOTHER EDGE UNIT BETWEEN (i, j) AND (i - 1, j)</p>	 <p>ADD (i, j) TO REGION₁. EXTEND EDGE LINE b₁ BY EDGE UNIT BETWEEN (i, j) (i, j - 1)</p>	 <p>CREATE NEW REGION₃ CONTAINING ONLY (i, j). CREATE TWO BOUNDARY EDGE LINES b₂ AND b₃</p>
 <p>MAKE b₁ AN OPEN CRACK. MERGE REGION₁ AND REGION₂ AND ADD (i, j) TO THE UNION</p>	 <p>ADD (i, j) TO REGION₂ AND ADD TO BOUNDARY LINE b₁ ANOTHER EDGE UNIT BETWEEN (i, j) AND (i - 1, j)</p>	 <p>ADD (i, j) TO REGION₁ EXTEND EDGE LINE b₁ BY EDGE UNIT BETWEEN (i, j) (i - 1, j)</p>	 <p>CREATE NEW REGION₃ CONTAINING ONLY (i, j). CREATE TWO BOUNDARY EDGE LINES b₂ AND b₃</p>
 <p>IF A ≠ B, THEN MERGE A AND B AND ADD (i, j) TO THE UNITIZED REGION. COMBINE BOUNDARY LINES b₁ AND b₂</p>	 <p>IF A = B, THEN COMBINE BOUNDARY LINES b₁ AND b₂ AND MAKE b₃ A CLOSED CRACK IF A ≠ B b₃ IS CURRENTLY A BOUNDARY</p>	 <p>IF A = B, THEN COMBINE BOUNDARY LINES b₁ AND b₂ AND MAKE b₃ A CLOSED CRACK IF A ≠ B b₃ IS CURRENTLY A BOUNDARY</p>	 <p>START NEW REGION, REGION₄ AND TWO BOUNDARY LINES b₃ AND b₄</p>

Fig 9 The different region growing decisions



THE 3 OPTIONS TO EXTEND AN OPEN CRACK AND THE CORRESPONDING ASSUMPTION ON DISTRIBUTING

FIG 10 The three options to extend an open crack and the corresponding assumption on distributing

Merging of two regions may always result in transformation into a crack of a previously common boundary of the two regions. In general each operation of the region grower is fairly elaborate. The data structure used is not described in this paper, but it is essentially the same data structure described in [24] with slight modification to include edge line representation through chain encoding.

This one-pass algorithm is local and requires relatively small core resident data. However it does not create maximal regions with respect to directionality of the region growing. On the other hand, it is relatively simple and fast when other algorithms are considered. The maximality problem may be easily corrected if backup is allowed. Note also that the threshold T plays a very small role in defining the output of the algorithm.

9. Simplification of the Result of Basic Region Growing

There are two straightforward options for simplifying the output of the one-pass region grower: (1) take all regions that are too small to be interesting and melt them into their closest neighbor (the distance between two regions will be defined later in the paper); (2) take all short cracks that are weak (strength of the edge line will be defined later) and delete them. Of course the threshold below which a crack is weak and a region is small is a function of how much we want to elaborate the task of the image analysis and is defined heuristically. In fact, in the current implementation all cracks are deleted since the edge operator is sensitive enough for our purposes.



FIG. 11 Original

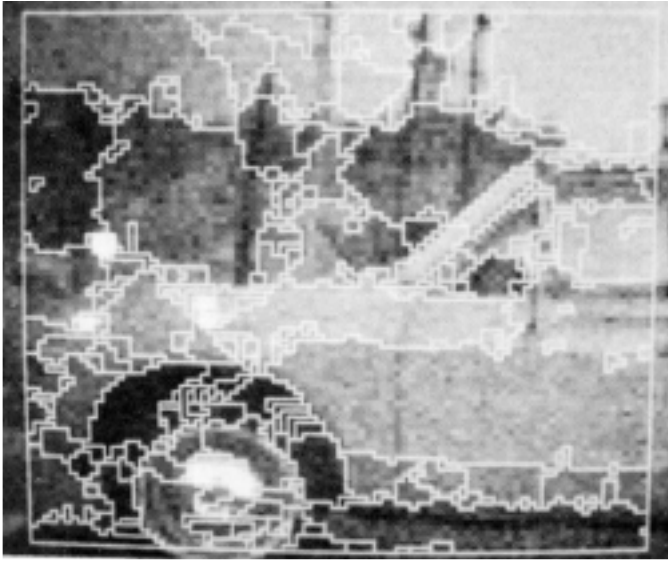


FIG. 12. Region growing based on proposed region grower (Section 7) using the edge evaluation of Section 3 with default thresholds

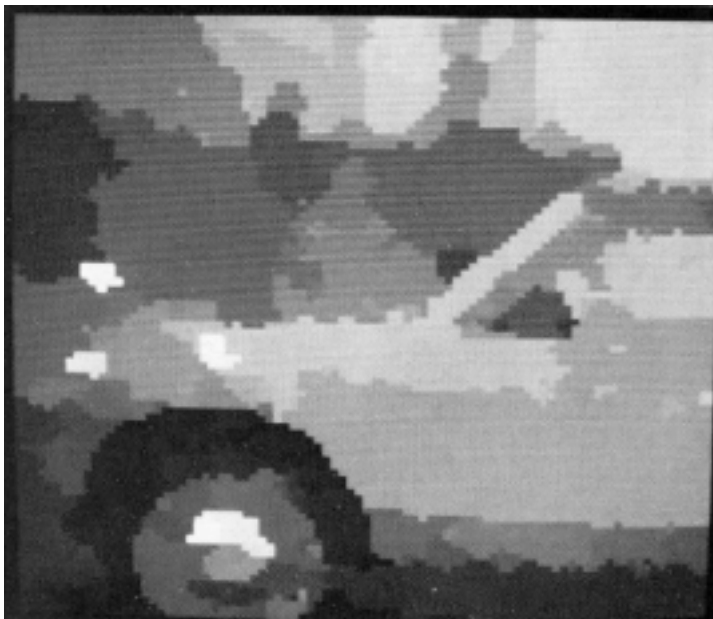


FIG 13. Reconstruction of the picture from the information contained in the regions of Figure 12

10. Growing Open Cracks Into Closed Cracks

At the present time we just obtain the cracks within regions and mark them as such. In the future one may look for ways of closing them to define finer regions. One possible way of closing open cracks is to grow them in length from their open end until the

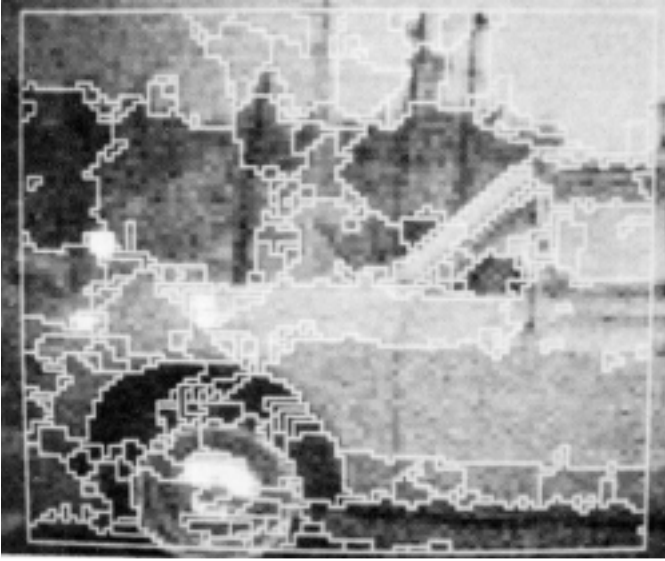


FIG. 14 Melting regions from Figure 12 using default threshold on second pass

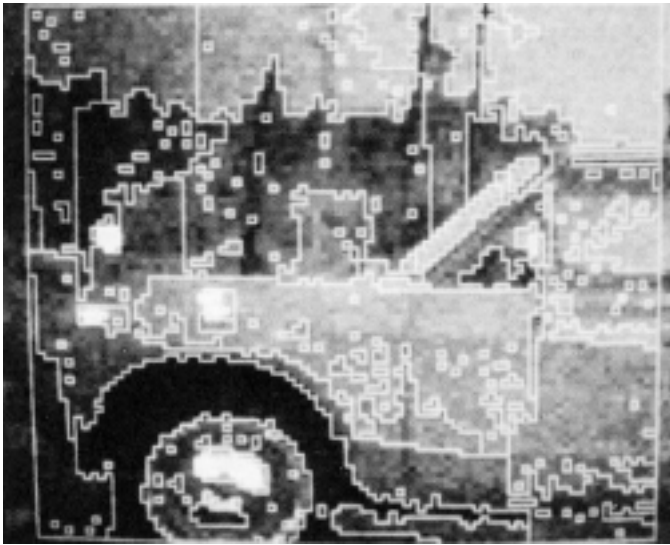


FIG. 15 Growing regions with the only constraint that maximum range of gray level in a region is less than 20 (20 selected as best for that image)

extended edge line meets already existing edge lines and closes. On the open end in each step there are three choices of where to extend the edge line: go straight ahead, turn left, or turn right. The decision as to which direction to take will minimize the cost of closing the open crack, where the cost is defined heuristically. One possible choice is as follows:

Given the original crack, define two distributions that describe the properties on either side of the crack, P_{D_1} and P_{D_2} . The cost of adding an edge unit will be the maximum likelihood ratio between the two assumptions:

H_0 : the two sides of the edge unit belong to the same side of the crack (the best choice between D_1 on both sides of the extension and D_2 on both sides);

H_1 : there is a different distribution on either side chosen according to geometrical

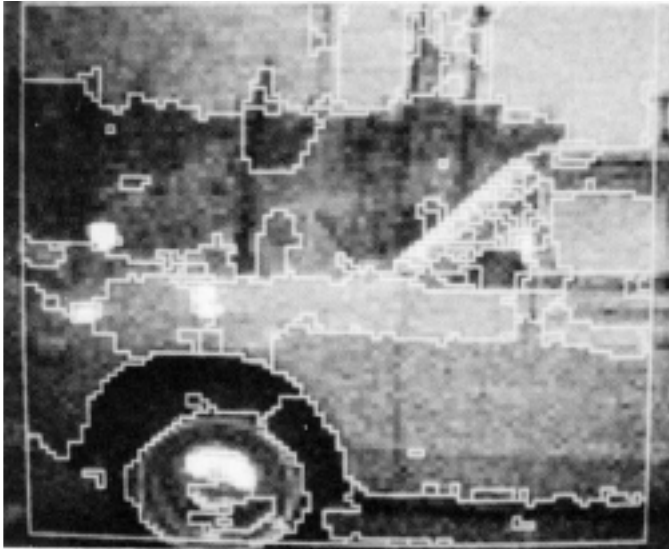


FIG 16 Region growing using $|\sum_{i=1}^n (X_i - Y_i)|$ as the edge value with the proposed one pass region grower Threshold which was selected manually as optional to that picture was set to $5 \cdot n$



FIG 17 Original

constraint (Figure 10).

$$\text{Cost} = P_{H_0} / P_{H_1}.$$

Note that since the cost function is additive it can be used in conjunction with the shortest path algorithm [16, Ch. 3] to find the nearest (least expensive) path to a closing

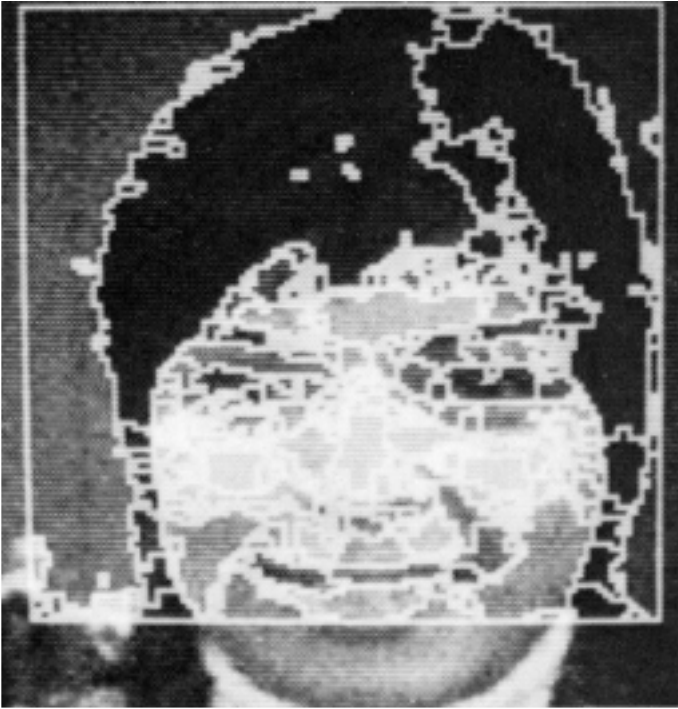


FIG 18 Region growing default parameters (like Figure 12)



FIG 19 Reconstruction of Figure 18

edge unit. Reference [16] describes an application of the shortest path algorithm to edge extension under simpler constraints. This technique is immediately applicable to our different cost function and the different termination condition on the path.



FIG 20 Melting second pass with default threshold applied to Figure 18

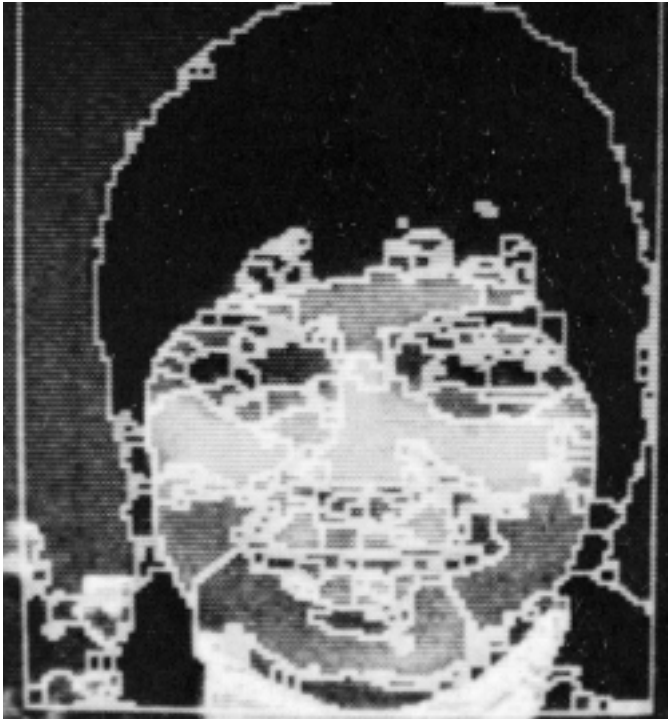


FIG 21 Region growing with edge values set to $\sum_{i=1}^n (X_i - Y_i)$ with threshold set to $8 \cdot n$



FIG. 22. Region growing so that maximum range of gray levels within a region is less than 25

11. *Breaking a Region Into Two Around a Crack*

An alternative approach to breaking a region into two regions to make the crack into a part of a boundary is to use special purpose region growing. Assume that there is a crack in a regular gray level picture ($V(i, j) \in R_1$) with readings with mean μ_1 and variance σ_1^2 on a small neighborhood on one side of the crack and mean μ_2 and variance σ_2^2 on the other side. Assume that the crack is inside region R ; then we can break the points in R into two classes, C_1 and C_2 :

$$C_1 \triangleq \left\{ (i, j) \mid \frac{1}{\sqrt{(2\pi) \cdot \sigma_1}} \cdot e^{-\frac{1}{2}(V(i, j) - \mu_1)^2 / \sigma_1} \leq (1/\sqrt{(2\pi) \cdot \sigma_2}) \cdot e^{-\frac{1}{2}(V(i, j) - \mu_2)^2 / \sigma_2} \right\}, C_2 \triangleq R \sim C_1.$$

Then we would expect C_1 to be on the first side of the crack and C_2 on the second side of the crack. Unfortunately it may turn out that C_1 or C_2 is not pathwise connected. As a result one of the connected components that border on the crack should be picked out. A more heuristic approach is to grow a region around each of the two sides of the crack, and to stop when a new point has a neighborhood that is more likely to belong to the other side. Then take the smaller of the two regions resulting and make it C_1 ; then C_2 will be $C_2 = R - C_1$.

This algorithm can be used also to allow flexible human intersection in analyzing the scene. This can be done by allowing the operator to use cursor sample points of the two subregions of a region. The machine then defines the separating features of the two subregions and carries out the segmentation. [12] and [23] describe systems that automatically select the distinguishing features of subregions of a given region. The local structure around a crack provides the distinguishing features of the two sides, and implementing an automatic system for doing that should be relatively easy. Currently this option is not used in our system, but descending from down region to subregion is of great potential value.

12. Merging Regions

This basic region grower utilizes local detection procedures. Better decisions are achievable (at least theoretically) by using more global information. The problem is how to use this additional information and still keep the program lean and fast. Research in this area has been reported [24]. Our basic approach is to be oversensitive on the local pass and as a result to oversegment the picture. But then we take the output data (which is simple relative to the original picture) and simplify it. We take pairs of regions with common boundaries and merge them into one. To do that reliably, a confidence value that measures the confidence that the pair of regions are different is computed. Then we iteratively select the pair of regions with the lowest confidence of being different in the current structure, merge them, and update the structure. The confidence is dependent on two factors: (1) edge line strength (on the common boundary of the two regions), and (2) the difference of the properties inside the two regions. Both of these values are computed on the basis of assumptions similar to those used in the edge confidence evaluation. For instance, if we assume gray level readings and let $x_i, i = 1, n$, be the readings on one region and $x'_i, i = 1, m$, be the readings at the other, then the second factor is

$$CONFIDENCE = V_0^{m+n}/V_1^n \cdot V_2^m,$$

where

$$\begin{aligned} (1) \quad \mu_0 &= \left(\sum_{i=1}^n X_i + \sum_{i=1}^m X'_i \right) / (m+n), & (4) \quad V_1 &= \left(\sum_{i=1}^n (X_i - \mu_1)^2 \right) / n, \\ (2) \quad V_0 &= \left(\sum_{i=1}^n (X_i - \mu_0)^2 + \sum_{i=1}^m (X'_i - \mu_0)^2 \right) / (m+n), & (5) \quad \mu_2 &= \left(\sum_{i=1}^m X'_i \right) / m, \\ (3) \quad \mu_1 &= \left(\sum_{i=1}^n X_i \right) / n, & (6) \quad V_2 &= \left(\sum_{i=1}^m (X'_i - \mu_2)^2 \right) / m. \end{aligned}$$

Results using only this factor are shown below. In [24] the local boundary properties are used to compute the edge values. The merging is stopped when the weakest boundary strength is more than a given threshold

13. Results

The suggested one-pass region growing algorithm driven by edge values was implemented on the General Automation SPC-16/75 minicomputer of the JPL robotics lab. The input picture is digitized into 256 gray levels from the black and white video signal of a Cohu camera. The noise variance is 2, measured from repetitious readings of the same point in a sequence of images. A Ramtek display unit is interfaced with the minicomputer and is used to display the digitized picture in green. Boundary lines of regions are displayed in red over the original picture for performance evaluation.

All cracks are currently ignored. The threshold below which the edge value is truncated to 0 was fixed at 2000 in all the examples below. A system to set the threshold automatically to allow only 5 percent of the points of the image to have value over the threshold was scrapped in favor of an absolutely fixed threshold.

The output of the first pass is then passed to a region merger that reduces the number of regions also with a default fixed threshold (merge till log (confidence) is greater than or equal to 20). The compute time for a 200×200-pixel picture is approximately a minute for a program that is highly inefficient because of debugging aids.

The results shown in Figures 11-22 are encouraging. We believe that the use of planar fits (the gradient edge detector instead of the step edge detector) and the dynamic use of region features as they grow to upgrade performance of the region grower will result in

even better performance. We found the region growing algorithm an important tool in scene analysis [25], and look forward to improving its performance.

Comparison of performance of the suggested region growing approach with two others is given. The first alternative is the one-pass region growing described in Section 7, i.e. the use of algorithms that grow regions so that the difference between the maximum gray level reading and the minimum gray level reading in a region is less than a predefined threshold. The second alternative uses the same region growing where the edge value is taken to be $|\sum_{i=1}^n (X_i - Y_i)|$, where X_i are the readings from one neighborhood and Y_i are those from the other. The neighborhoods are identical to those of the proposed edge operator. In the best case the two alternatives performed similarly to the proposed region grower. However, the threshold setting had to be adjusted manually between pictures until the performance became comparable to the proposed (adaptive) edge detector.

REFERENCES

1. BAJCY, R. Computer identification of textured scene. AIM-180, STAN-CS-72-321, Stanford U., Stanford, Calif., 1972.
2. BARROW, H., AND POPPLESTONE, R. Relational description in picture processing *Machine Intelligence 6*, B Meltzer and D Michie, Eds, Edinburgh U Press, Edinburgh, 1971, pp 377-396.
3. BERTHOLD, K., AND HORN, P. The Binford-Horn Line-Finder Memo No 285, M I T Artificial Intelligence Lab, M I T, Cambridge, Mass, Dec 1973
4. BINFORD, T., AND HERSHKOVITZ, A. On boundary detection Memo No 183, M I T Artificial Intelligence Lab., M I T, Cambridge, Mass, Dec 1970.
5. BRICE, C., AND FENNEMA, C. Scene Analysis Using Regions. *J. of Art Intell 1* (1970), 205-226.
6. CARTON, E.J., ET AL. Some basic edge detection techniques TR-277, Computer Science Center, U. of Maryland, College Park, Md, Dec 1973
7. DAVIS, L. A survey of edge detection techniques C S TR-273, U of Maryland, College Park, Md, 1973
8. FUKUNAGA, K. *Introduction to Statistical Pattern Recognition* Academic Press, New York, 1972.
9. GRIFFITH, A. Mathematical models for automatic line detection. *J ACM 20*, 1 (Jan. 1973), 62-80
10. GRIFFITH, A. Edge detection in simple scenes using a priori information *IEEE Trans. on Computers C-22* (April 1973), 371-381
11. HARLOW, C.A., AND EISENBEIS, S.A. The analysis of radiographic images. *IEEE Trans on Computers C-22* (1973), 678-689
12. HOROWITZ, S., AND PAVLIDIS, T. Picture segmentation by a direct split and merge method Proc 2nd Joint Int Conf of Pattern Recognition, IEEE Pub 74CHO-885-4C, Aug 1974, pp. 424-433
13. HUECKEL, M.H. A local visual operator which recognizes edges and lines *J. ACM 20*, 4 (Oct 1973), 634-647
14. LIPKIN, B., AND ROSENFELD, A. *Picture Processing and Psychopictorics* Academic Press, New York, 1970, pp 382-383.
15. MARTELLI, A. Edge detection using heuristic search method. *Computer Graphics and Image Processing* (1972), 169-182
16. NILSSON, N. *Problem Solving Methods in Artificial Intelligence*. McGraw-Hill, New York, 1971
17. PINGLE, K., AND TENENBAUM, J. An accommodating edge follower. Proc International Joint Conf on Artificial Intelligence, British Computer Soc, London, England, Sept 1971, pp 1-7
18. ROSENFELD, A. Picture processing by computer *Computing Surveys 1*, 3 (Sept 1969), 147-176.
19. ROSENFELD, A. Progress in picture processing: 1969-1971 C S TR-176, U of Maryland, College Park, Md, Jan 1972
20. ROSENFELD, A. Picture processing: 1972 C S TR-217, U. of Maryland, College Park, Md, Jan. 1973
21. SHIRAI, Y. A step toward context-sensitive recognition of irregular objects. *Computer Graphics and Image Processing 2*, 3 (Dec 1973), 298-307
22. TENENBAUM, J. Accommodation in computer vision EE Th, Stanford U., Stanford, Calif., 1970
23. TOMITA, F., ET AL. Detection of homogeneous regions by structural analysis Proc 3rd Int Joint Conf on Artificial Intelligence, Aug 1973, pp 364-371
24. YAKIMOVSKY, Y. Scene analysis using a semantic base for region growing STAN-CS-73-380, Stanford U., Stanford, Calif., June 1973
25. YAKIMOVSKY, Y. On the recognition of complex structures Proc. of 2nd Joint Int Conf of Pattern Recognition, IEEE Pub 74CHO-885-4C, Aug. 1974, pp 345-353

RECEIVED JUNE 1974, REVISED MARCH 1976