BOUNDARY AND OBJECT DETECTION IN REAL WORLD IMAGES

Yoram Yakimovsky

Jet Propulsion Laboratory
California Institute of Technology
4800 Oak Grove Dr.
Pasadena, California 91103

Abstract

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

I. 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, tracing of objects in sequence of pictures for control systems, image enhancement, data reduction, and various other applications. References 1, *Z*, and 3 comprise a good survey of research and applications in image processing and picture analysis.

Most researchers of picture analysis 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 Ref. 4 for an example of texture image analysis which does not make these assumptions).

The work on automatic location of objects in digitized images has split into two approaches: (1) edge detection and edge following vs (2) region growing. Edge detection meant applying in different points over the picture local independent operators to detect edges and then using algorithms to trace the boundaries by following the local edge detected. A recent survey of literature in this area is given in Ref. S. The region growing approach was to use various clustering algorithms to grow regions of almost uniform local properties in the image. (See Refs. 6-9 for typical applications.) More detailed references will be given later.

In this paper the two approaches are combined to complement each other. The end result is a more powerful mechanism to do the job of segmentation pictures into objects. We developed a new edge detector and combined it with new region growing techniques to locate objects and thereby resolved the confusion that has resulted for regular edge following when more than one isolated object on a xmiform background is in the scene (see Ref. 10).

The contributions of this report are *the* following:

(1) A new and "optimal" (given certain assumptions) edge detector *is presented.*

(2) A simple one-pass *region growing algorithm* which was implemented on a mini-computer. It utilizes the edge detector output.

(3) The application of path generator algorithms and "shortest path algorithms" to do the boundary following so as to close open edge lines into boundaries around regions.

(4) Special-purpose region growing intended to close open edges (cracks).

(5) A special clustering algorithm which simplifies the region structure resulting from application of (1) through (4).

II. Definition of Terms

The input is expected to be in matrix form $\vec{V}(i,j)$ $i = 1,\ldots,N$ $j = 1,\ldots,M$, where $\vec{V}$ is a vector in R^n, n is a function of the sensory system, usually 1 (gray level picture), or 3 (color or x, y, z coordinates of 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 Fig. 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_1^1, e_2^1, e_3^1 in Fig. 1 may exist or not.

Two points on the grid (I, J) and (K, L) are said to be in the same region if there is a path sequence $(i_1, j_1),\ldots,(i_n, j_n)$ 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 $m = 1,\ldots n - 1$ 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 R1 and region R2 is the maximal sequence of adjacent edge units such that each edge unit in the sequence is between two matrix points, one belonging to R1 and the other to R2. It is possible that an edge line is inside a region (R1 = R2).

An edge line which is between two different regions is called a boundary. An edge line which 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 which terminates at both ends on another edge line. For instance, cracks will appear when an object is smoothly disappearing into the background on one side and has detecable discontinuity on the other side (Fig. 2).

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

HI. 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 I l,j), it should return a value which will measure the confidence that there is an edge between (i, j) and (i + l,j). Since we work with noisy input to achieve reliability, the operator must look at two 2-dimensional (2-D) neighborhoods *N*\ and N2 to obtain a reliable value. Neighborhood Nj will include (i,j) and a few adjacent points; N2 includes (i + 1, j) and a few adjacent points; and Nj *(~\ N%* 0. As a result the value returned will measure the confidence that the neighborhoods belong to images of different objects.

Kdge detection is actually composed of three components: (!) choosing the proper neighborhoods, (2) the measurements of differences between image structures in the two neighborhoods, and (i) locking on the exact position of the edge. Discussion of each of these steps follows.

IV. Measuring Differences in Structure Between Two Neighborhoods

Any technique which measure structural differences must make some assumption (explicitly or implicitly) on the structure of an edge vs the area inside a region. lUnford and Hershkovitz (Ref. 22) suggested three possible ideal edges defined by the intensity profile on a normal-to-the-e.dge line (Fig. 3).

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

Ho- The readings in Nj and N2 are taken from the same object.

\\\: The readings in Nl are taken from one object and in N2 from another object.

Neighborhoods N1 and N2 are the neighborhoods mentioned in the previous section, and the decision ,vs to how to choose them will be described in the next section.

An optimal (best for its size) decision between ! It) and II] will utilize the maximum likelihood ratio as follows: L<et Pn be the maximum likelihood estimate of the structure (reading in N\ and N2), given that Ho is true, and let Pi be the max imum likelihood estimate of the structure, assuming *H*\ is true. Then,

$$\text{Choose } H_1 \text{ when } \frac{P_1}{P_0} > K$$

$$\text{Choose } H_0 \text{ when } \frac{P_1}{P_0} < K$$

If $P_1/P_0 = K$ choose at random.

This decision will be optimal for its size (see the Neyman-Pearson Test in Ref. 11, pg. 55); hence, if our assumptions are valid, we have an ideal edge detector, given only readings in N_1 and N_2. (We will deal with gaussian probabilities; hence we will 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 edges of types (a) and (b) in Fig. 3.

EXAMPLE 1

Assume that the edges and surfaces will be of type A, with added white noise which is object-dependent. Then H_0 and H_1 will 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 $N(\mu_1, \sigma_1)$; and the readings on N_2 are taken from normal distribution $N(\mu_2, \sigma_2)$; (μ_1, σ_1) need not be equal to (μ_2, σ_2).**

To apply the maximum likelihood ratio principle we need to find a maximum likelihood estimate for (μ_0, σ_0) (μ_1, σ_1) and (μ_2, σ_2). Given $(x_1, \cdots, x_n)$ readings taken from a normal distribution with unknown (μ, σ), the maximum likelihood estimates are

$$\bar{\mu} = \frac{\sum_{i=1}^{n} x_i}{n}$$

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

Let P_{max} the probability of getting those readings assuming they are taken independently from normal distribution $N(\bar{\mu}, \bar{\sigma})$ then

$$P_{max} = P_{(\bar{\mu}, \bar{\sigma})}(x_1, \cdots, x_n)$$

$$= \frac{1}{(\sqrt{2\pi}\cdot\bar{\sigma})^n}\cdot e^{-\frac{1}{2\bar{\sigma}^2}\cdot\sum_{i=1}^{n}(x_i-\bar{\mu})^2}$$

$$= \frac{1}{\left(\sqrt{2\pi}\cdot\sigma\right)^n}\cdot e^{-\frac{n\sigma^2}{2\sigma^2}}$$

$$= \left(\frac{1}{\sqrt{2\pi}^n}\cdot e^{-\frac{n}{2}}\right)\cdot\frac{1}{\sigma^n}$$

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

$$\mu_1 = \frac{\sum_{i=1}^{m} x_i}{m}$$

$$\sigma_1^2 = \frac{\sum_{i=1}^{m}(x_i-\mu_1)^2}{m}$$

$$P_1 = \left(\frac{1}{(2\pi)^{\frac{m}{2}}}\cdot e^{-\frac{m}{2}}\right)\cdot\frac{1}{\sigma_1^m}$$

On N_2,

$$\mu_2 = \frac{\sum_{i=1}^{n} y_i}{n}$$

$$\sigma_2^2 = \frac{\sum_{i=1}^{n}(y_i-\mu_2)^2}{n}$$

$$P_2 = \left(\frac{1}{(2\pi)^{n/2}}\cdot e^{-\frac{n}{2}}\right)\cdot\frac{1}{\sigma_2^n}$$

and on N_1 combined with N_2,

$$\mu_0 = \frac{m\mu_1 + n\mu_2}{m+n}$$

$$\sigma_0^2 = \frac{m\sigma_1^2 + n\cdot\sigma_2^2 + m(\mu_0-\mu_1)^2 + n(\mu_0-\mu_2)^2}{m+n}$$

$$P_0 = \frac{1}{(2\pi)^{\frac{m+n}{2}}}\cdot e^{-\frac{m+n}{2}}\cdot\frac{1}{\sigma_0^{m+n}}$$

$$\frac{P_1^2\cdot P_2^2}{P_0^2} = \frac{\left(\sigma_0^2\right)^{m+n}}{\left(\sigma_1^2\right)^m\cdot\left(\sigma_2^2\right)^n}$$

$$= (\text{LIKELI-HOOD RATIO})^2$$

Here we assume that when assumption H1 holds the readings on the two neighborhoods are independent. That is, the maximum probability assuming *H1* is the product of *P1* and P2 To save computation of square roots we work with 2 instead of the

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 easily In practice, we computed the variance of noise in the hardware by sampling over time the same points in static scenes. All variances were forced to be at least the hardware noise so as to prevent divisions by zero in pathological cares.

At this point it may be worthwhile to compare our approach with that of Ref. 12. 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, Ref. 12 assumes a priori deterministic classification of all possible idealized noise-free structures to 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 on edge.

It should be mentioned that other statistical techniques were used for edge detection as in Ref 22 and Ref. 18, but none of the edge detectors which appeared in the literature used the maximum likelihood test for edge value.

EXAMPLE 2

Here we assume that each matrix point V(i, j) is a 3-dimensional vector (x,y,z). Actually the raw readings are just distance R(i,j), but to avoid a strong dependency on the sensory position, (i,j,R) are used to compute (x,y,z). This is the form of input read from a device which measures distances to surface (such as radar or devices which measure the time of flight of laser beams to an object). The i, j corresponds to vertical and horizontal steps in the scanning angle. The two adjacent neighborhoods on the matrix Nl and N2 have readings $(x_1, y_1, z_1), \cdots,$ (x_n, y_n, z_n) in N_1 and $(x1^1, y1^1, z_1^1), \cdots,$ (x_m^1, y_m^1, z_m^1) in N2. We assume that objects are almost planar locally with added white noise with mean to position readings. That is, if we read $(x_1, y_1, z_1), \cdots, (x_n, y_n, z_n)$ in a small neighborhood on an object we have a,b,c,d,<r such that

$$a^2 + b^2 + c^2 = 1$$

and

$$ax_i + by_i + cz_i + d + N(0,\sigma) = 0 \qquad i = 1, \cdots, n$$

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 $(a_0, b_0, c_0, d_0, \sigma_0)$

$$a_0x + b_0y + c_0z + d_0 + N(0, \sigma_0) = 0$$

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

$$a_1^2 + b_1^2 + c_1^2 = 1$$

$$a_2^2 + b_2^2 + c_2^2 = 1$$

$$i = 1, \cdots, n \qquad a_1x_i + b_1y_i + c_1z_i + d_1 + N(0, \sigma_1) = 0$$

$$i = 1, \cdots, m \qquad a_2x_i^1 + b_2y_i^1 + c_2z_i^1 + d_2 + N(0, \sigma_2) = 0$$

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

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

$$= \min_{\substack{a,b,c,d \\ a^2 + b^2 + c^2 = 1}} \sum_{i=1}^{n} (ax_i + by_i + cz_i + d)^2$$

and

$$\sigma_1^2 = \frac{V_i}{n}$$

Solving for the optimal (a1, *b1, c1, d1)* is a relatively straightforward process. Once they are found, the maximum likelihood estimate for N1 will be

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

Hence, we have the expression which tests for an edge. It is of the following form: If

$$\frac{V_0^{m+n}}{V_1^n \cdot V_2^m} \geq K^2$$

decide for H1, otherwise H0.

Note that (x,y,z) may be replaced by (i,j,g) in regular black and white pictures, in which case we will have a regular picture edge operation which will be able to handle edges of type B in Fig. 3. Somewhat similar applications were reported in Ref. 20 and Ref. 22 for dection of gradient edges Fig. 3(b). This edge operator has not yet implemented on our system.

V. Locking on a Detected Edge

Computing the edge value is not usually sufficient to decide where to put the edges. The values that are computed usually look like the ones in Fig. 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 Ref. 23, page 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* problems here is that, because of computing time constraints, it was impossible to find a global optimum for edge lines using all available data, and it was necessary to use only local information for evaluating the edge units in this level. In our system, the decision as to where exactly to put the edge was left for the region grower (see below). To demonstrate the possible 2-D ambiguity, see Fig. 7.

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 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.

VI. Region Growing

The output of an 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 + l,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 in order to recognize features. One way of achieving that is to use a region grower which will outline objects by clustering points into regions. This approach was used in the past for picture analysis. Recent works along that line are Refs. 6, 7, 8, and 9- The basic conclusion of those works is that without using semantic information, which is the knowledge on the subject of the picture, clustering cannot create perfect outlines. More recent work, Ref. 24 and 11, pg. 324, introduced new tech' niques of clustering which provide more flexibility and may upgrade clustering performance for images .

Here we introduce a new algorithm for clustering based on search for "valleys" of edge values in a picture If random access is allowed, a relatively simple algorithm which starts from local minima of edge values and climb up can be implemented. Due to lack of storage capacity on our mini-computer, and in an attempt to use data as it digitized from the video signal sequentially, a one-pass algorithm to generate regions corresponding to valleys was implemented. To our knowledge, this is the first time that this approach is 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" which was used in various works like Refs. 10, 16, 19, and 21. The basic idea in edge following is to detect a discontinuity and trace it and this way define edge lines. Unfortunately, the works in edge following suffers from lack of an effective way of tying regions properties into their decision processes and output.

Let us start by describing a one-pass algorithm which the edge data into data structures of regions, boundaries, closed cracks and open cracks, and creates, as byproducts, 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 ease 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. It will be

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

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

(2) **Reg(i, j) will be the region to which the point (i, j) belong. (Reg(i, j) is not defined to all points until the program is finished.)**

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

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) $Val(Reg_1) = \underbrace{\text{Min}(Val(i,j))}_{(i,j)}$

$Reg(i,j) = Reg_1$

(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 non-decreasing 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 $(x_1, x_2, \cdots, x_n)$ such that

(a) $x_1 = P, \quad x_n = Q$

(b) $Reg(x_i) = Reg(P), \quad i = 1, \cdots, n$

(c) x_i adjacent to x_{i+1}, $d(x_{i+2}, x_{i+1}) \ge d(x_{i+1}, x_i)$

(d) $d(x_n, x_{n-1}) = Val(x_n)$

We say if such a path exists that 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 which does not cross a ridge of edge values.

VII. Algorithm Description

The program scans the image from left to right, line by line. That is, the scanning is such that when point (i,j) is processed, the program already worked on all points (i_1, j_1) such that $(j_1 < j)$ or $(j = j_1$ and $i_1 < i)$.

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

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

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

```
Begin

Boolean Good-Down1, Bad-Down1, Up1,
Good-Down2, Bad-Down2, Up2;
Good-Down1 ← (d((i, j), (i, j-1)) ≤ Val(i, j-1))∧
   ((Val(i, j-1) ≤ Val(R1))

   Comment: Good-Down1 is true if point (i, j) is
   going to become a new minimum for R1 (the
   region above), and it is adjacent to an old
   minimum; hence, any point of R1 reachable
   from the old adjacent minimum will be
   reachable from the new;

Bad-Down1 ← (d((i, j), (i, j-1)) < Val(i, j-1))∧
   ((Val(i, j-1) > Val(R1)))

   Comment: This variable is true if (i, j) is not
   reachable from all minima of R1 going through
   (i, j-1);

Up1 ← d((i, j), (i, j-1)) ≥ Val(i, j-1)

   Comment: This variable is true if point (i, j)
   is reachable from any minimum of R1 by con-
   tinuing the path that leads from that minimum
   to (i, j-1);

Good-Down2 ← (d((i, j), (i-1, j)) ≤ Val(i-1, j))∧
   ((Val(i-1, j) ≤ Val(R2)))

   Comment: This variable is true if point (i, j)
   is going to be a new minimum for R2 (the
   region minimum, to the side) and is adjacent
   to an old minimum of R2; hence any point
   reachable from the adjacent old minimum will
   be reachable from (i, j);

Bad-Down2 ← (d((i, j), (i-1, j)) < Val(i-1, j))∧
   (Val(i-1, j) > Val(R2))

   Comment: This variable is true if (i, j) is not
   reachable from all minima of R2 through
   (i-1, j);

Up2 ← d(i, j), (i-1, j) ≥ Val(i-1, j)

   Comment: This variable is true if point (i, j)
   is reachable from any minima of R2 by con-
   tinuing the path that leads from that minimum
   to (i-1, j);

If Good-Down1 ∧ Good-Down2 then D1 ← D2 ← true
else
If Good-Down1 ∧ Bad-Down2 then begin D1 ← true;
D2 ← false; end else if Good-Down1 ∧ Up2 then
begin

   if d((i, j), (i, j-1)) ≥ d(i(i, j), (i-1, j))
   then D1 ← D2 ← true
   else begin D1 ← true;      D2 ← false; end;
                  end

else if Bad-Down1 then begin if Good-Down2 V
Up2 then begin

   D1 ← false
   D2 ← true
   end
   else begin
   D1 ← false
   D2 ← false; end
   end

else if Up1 ∧ Good-Down2 then begin

   if d((i, j), (i-1, j)) ≥ d((i, j), (i, j-1))
   then D1 ← D2 ← true
   else begin D2 ← true;      D1 ← false; end
                  end

else if Up1 ∧ Up2 then, if R1 = R2 then
D1 ← D2 ← true else

   if d((i, j), (i-1, j)) ≥ d((i, j), (i, j-1))
   then begin D1 ← true;      D2 ← false; end
   else begin D1 ← false;     D2 ← true; end
end

   Comment: In that case we must force that
   only one of D1 and D2 can be true; otherwise
   we cannot guarantee entrance through mini-
   mum value from all minima of both R1 and R2;

else if Up1 ∧ Bad-Down2 then begin D1 ← true;
D2 ← false; end

   Val(i, j)← ∞ ;
   if D1 then begin
   Val(i, j-1) ← Min(d((i, j), (i, j-1)), Val(i, j-1));
   Val(i, j) ← d((i, j), (i, j-1));
   Val(R1) ← Min(Val(R1), Val(i, j));
   end;
   if D2 then begin
   Val(i-1, j) ← Min(d((i, j), (i-1, j)), Val(i-1, j));
   Val(i, j) ← Min(Val(i, j), d((i, j), (i-1, j)));
   Val(R2) ← Min(Val(R2)), Val(i, j));
   end;
   If not (D1 V D2) then Val(Reg(i, j))← ∞ ;
```

The e_1 and e_2 (see Fig. 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 Fig. 9 for a brief description of the different cases.)

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: more than meets the eye. The data structure used is not described in this paper, but it is essentially the same data structure described in Ref. 9, 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 our criteria of path connectivity and reachability. The reason is the possible 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, in fact, the threshold T plays a very small role in defining the output of the algorithm.

VIII. 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 which 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 was sensitive enough for our purposes.

IX. Merging Regions

The basic region grower utilized local detection procedures. Better decisions are achievable (at least theoretically) by using more global information. The problem is how to allow this additional information and still keep the program lean and fast. Research in that area was reported (Ref. 9). Basically, our 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 original picture) and simplify it. We take pairs of regions with common boundaries and merge them into one In order to do that reliably, a confidence value which measures the confidence that the pair of regions are different is computed. Then, iteratively we pick 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) the magnitude of the change across the boundary 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, then let XJ i = 1, n be the readings on one region and xi' i = 1, m be the readings at the other, then the second factor will be:

$$\text{CONFIDENCE} = \frac{V_0^{m+n}}{V_1^{n} V_2^{m}}$$

where

$$(1) \qquad \mu_o = \frac{\sum_{i=1}^{n} X_i + \sum_{i=1}^{m} X_i'}{m+n}$$

$$(2) \qquad V_o = \frac{\sum_{i=1}^{n} (X_i - \mu_o)^2 + \sum_{i=1}^{m} (X_i' - \mu_o)^2}{m+n}$$

$$(3) \qquad \mu_1 = \frac{\sum_{i=1}^{n} X_i}{n}$$

$$(4) \qquad V_1 = \frac{\sum_{i=1}^{n} (X_i - \mu_1)^2}{n}$$

$$(5) \qquad \mu_2 = \frac{\sum_{i=1}^{m} X_i'}{m}$$

$$(6) \qquad V_2 = \frac{\sum_{i=1}^{m} (X_i' - \mu_2)^2}{m}$$

Results using only this factor are shown below. In Ref. 9, the local boundary properties are used to compute the edge values. The merging is stopped when weakest boundary strength is more than a given threshold.

Results

The suggested one-paes region growing algorithm driven by edge values was implemented on G. A. SPC-16/75 mini-computer of the Jet Propulsion Laboratory robotics lab. The input picture is digitized from black and white video signal of Cohu camera. The signal is digitized into 256 gray levels. The noise variance as measured from repetitious readings of the same point in a sequence of images is 2. A Ramtek display unit is interfaced to the mini-computer 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 to 2000 in all the examples below. A system to set the threshold automatically so as to allow only 5 percent of the points of the image to have value over the threshold was scrapped in favor of fixed absolutely threshold.

The output of the first pass is then passed to a region merger which reduces the number of regions also with default fixed threshold (merge till log (confidence) > 20). The compute time for a 200 x 200 picture is approximately a minute for a program which is highly inefficient because of debugging aids.

The results which are shown below are encouraging. We believe that use of planar fits (gradient edge detector instead of the step edge detector) and using features of region dynamically as they grow to upgrade performance of the region grower will result even better performance. We found the region growing algorithm as an important task to scene analysis (Ref. 25) and look forward improving its performance.

References

1. Rosenfeld, A., "Picture Processing by Computer," Computing Survey, Vol. 1, pp. 147-176, September 1969.

2. Rosenfeld, A., Progress in Picture Processing: 1969-1971, C. S. TR-176, University of Maryland, January 1972,

3. Rosenfeld, A., Picture Processing: 1972, C. S. TR-217, University of Maryland, January 1973.

4. Bajcy, R., Computer Identification of Textured Scene, AIM-180, STAN-CS-72-321, Stanford University, 1972.

5. Davis, L., A Survey of Edge Detection Techniques, C. S. TR-273~ University of Maryland, November 1973.

6. Brice, C., and Fennema, C., "Scene Analysis Using Regions," Journal of Artificial Intelligence. Vol. I, pp. 205-226, 1970.

7. Barrow, H., and Popplestone, R., "Relational Description in Picture Processing," Machine Intelligence, Vol. 6$_f$ pp. 377-396.

8. Harlow, C.A., and Eisenbeis, S.A., "The Analysis of Radiographic Images," IEEE Transactions on Computers, Vol. L-22, pp. 678-689. 1973

9. Yakimovsky, Y , Scene Analysis Using a Semantic Base for Region Growing, STAN-CS-73-380, Stanford Univeristy, June 1973.

10. Pingle, K., and Tenenbaum, J., "An Accommodating Edge Follower," Proceedings of the International Joint Conference on Artificial Intelligence, September 1971.

11. Fukunaga, K., Introduction to Statistical Pattern Recognition, Academic Press, 1972

12. Griffith, A., "Mathematical Models for Automatic Line Detection," Journal of the Association of Computing Machinery, Vol. 20, ppV*62-80, 1973."

13. Carton, E.J., et al., Some Basic Edge Detection Techniques, TR-277, University of Maryland Computer Science Center, December 1973.

14. Hueckel, M.H , "A Local Visual Operator Which Recognizes Edges and Lines," Journal of the Association of Computing Machinery, Vol. 20, pp. 634-647, 1973.

15. Nilsson, N., Problem Solving Methods in Artificial Intelligence, McGraw-Hill, 1971.

16. Martelli, A., Edge Detection Using Heuristic Search Method, Computer Graphics and Image Processing, pp. 169-182, 1972.

17. Tomita, F., et al., Detection of Homogeneous Regions by Structural Analysis, 3 IJCA1 Aug. 73, pp. 364-371.

18. Tenenbaum, J., Accommodation in Computer Vision, Stanford Univ., EE Thesis, 1970.

19. Griffith, A., Edge Detection in Simple Scenes Using Apriorl Information, I.E.E.E. Transaction on Computers, Apr. 1973, p. 371.

20. Shirai, Y., Computer Graphics and Image Processing, Vol. 2. Nr. 3, Dec. 1973, pg. 298.

21. Berthold, K., Horn P , The Binford-Horn Line-Finder, MIT, A.T. Memo 285, Dec. 1973.

22. Binford, T., Herskovitz, A., On Boundary Detection, MIT, A.I. Memo 183, Dec 1970.

23. Rosenfeld, A., Picture Processing and Psychopictorics, pg. 382-383.

24. Horowitz, S., Pavlidis, T., Picture Segmentation by a Direct Split and Merge Method, Proceeding of the 2nd Joint Int'l. Conference of Pattern Recognition, pg. 424.

25. Yakimovsky, Y., On the Recognition of Complex Structures, Proceedings of 2nd Joint Int'l. Conference of Pattern Recognition, pg. 345.

Figure 1 one pass region growing using maximum likelihood edge detector default threshold
VAR = 2 TR = 2000

Figure 2 merging regions using iteratively weakest boundary first with stopping criteria $(m+N)\log V_o - m\lg V_1 - n\lg V_2 \geq 20$

Figure 3 reconstructed image from regions in Figure 2

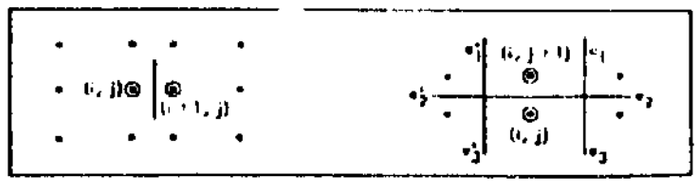


Figure 1.

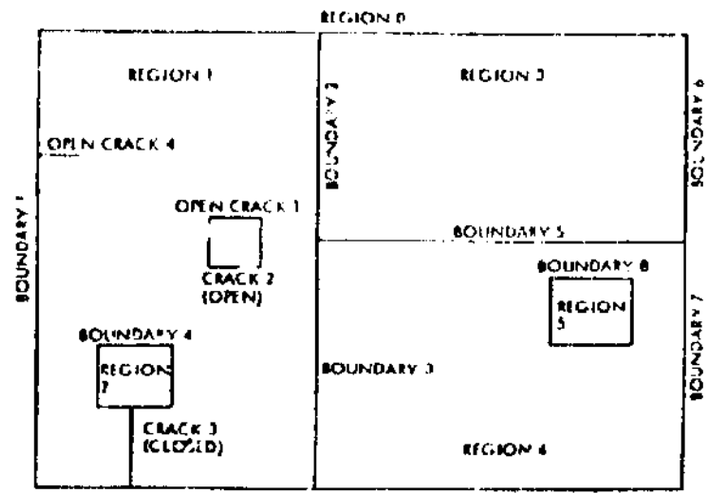
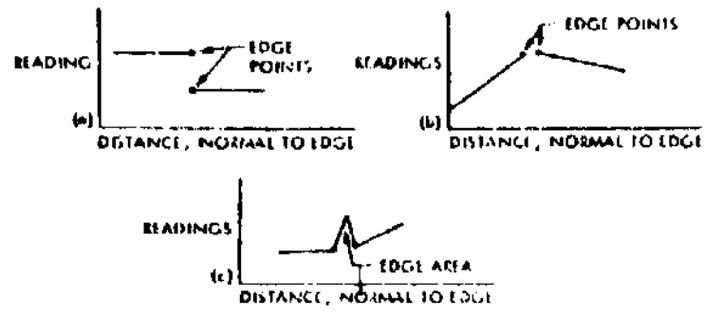


Figure 2.



(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).

Figure 3.

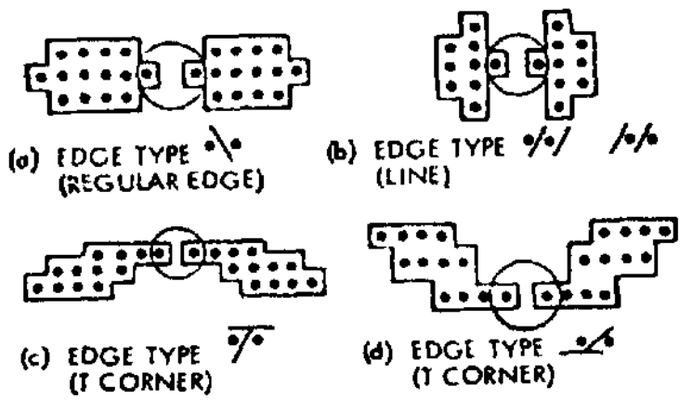


Figure 4.

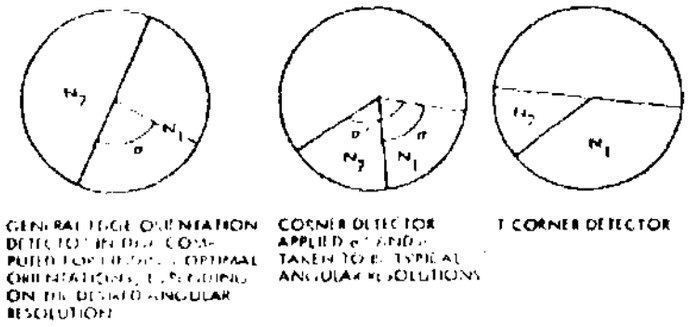


Figure 5.

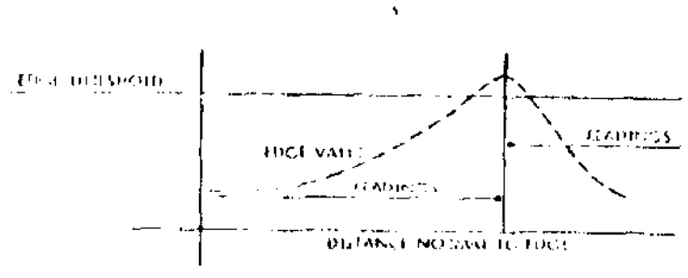
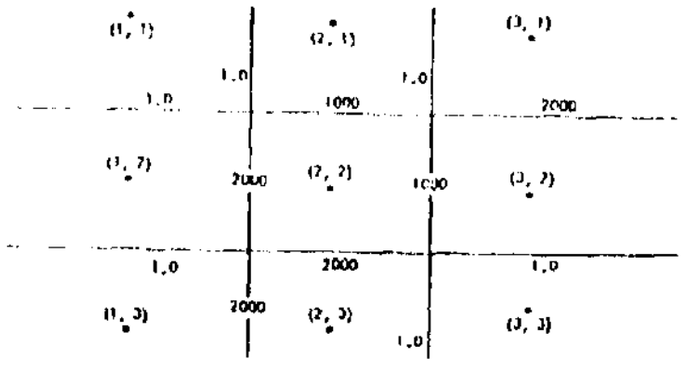


Figure 6.



THE (i, j) ARE POINTS NUMBER 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, 3) (3, 2) (2, 3) IN ANOTHER, BUT WHERE (2, 2) SHOULD BE IS TOTALLY AMBIGUOUS (ASSUMING THAT SINGLE POINT REGIONS ARE NOT ALLOWED)

Figure 7.

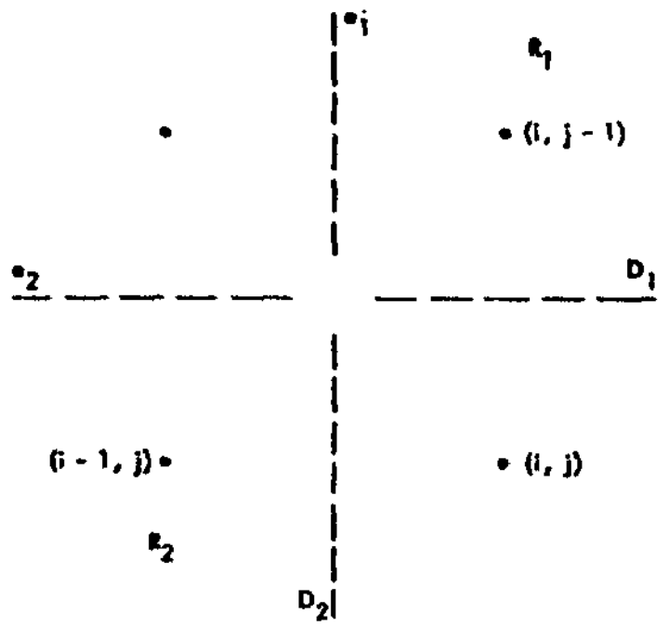
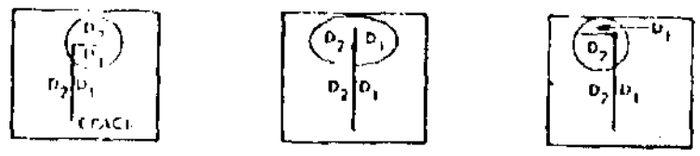


Figure 8.



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

Figure 9.