May 1969

A REGION-ORIENTED DATA STRUCTURE

by

Claude Fennema
Stanford Research Institute
Menlo Park, California

Claude Brice
Faculté des Sciences
Université de Toulouse
Toulouse, France

Artificial Intelligence Group

Technical Note 7

SRI Project 7494

This research was sponsored by the Advanced Research Projects Agency and was monitored by the Rome Air Development Center under Contract F30602-69-C-0056.

ABSTRACT

This technical note describes a region-based data structure that is easily obtained, lends itself to description of the data in a rich manner by a process of pointer reduction, and reduces combinatorial types of search by implicitly including positional information. The structure is also context-free and closed.

I INTRODUCTION AND PRELIMINARY DEFINITIONS

Considerable attention has been paid to developing a structure to describe regions and boundaries of a picture. To what extent should the topological or positional information be explicated,[1,2*] or rather easily retrieved from the structure?[3,4,5] Should a special coding system be developed for that purpose,[2,5,7] or should the current facilities of list-processing language be used? Our aim was to develop a structure that could be used for picture analysis and thus should be embedded in a scene-description program. It was important that topological information be easily retrieved although subject to change as the picture was processed by means of grouping regions. The picture representation described below is, therefore, oriented to provide this positional information in a flexible, yet immediately usable, form.

Initially, suppose the data is represented by an N x N matrix (array). P = (P(I,J)) of data points P(I,J) (picture points), and there is some kind of description function or property function

$$\mathcal{D}:P \rightarrow \pi D(I)$$

$$P(I,J) \rightarrow (RED,LIGHT,FAR,\ldots)$$

where each D(I) is a description space such as D_1 = color, D_2 = intensity, D_3 = range,..., etc.

The purpose here is to describe a structure whose data provides a means of describing the contents of the data in a rich, yet economical, way by collecting like data into classes. This structure also is so constructed to give great ease of manipulating both the original data and the data of the structure itself. The natural way to collect like data is to partition the array into equivalence classes based on the description function $\mathcal{D}$.

*References are listed at the end of the body of this technical note.

One such meaningful partition would be the partition of matrix P by the relation

$$(P(I,J) \sim P(K,L)) \text{ IFF } \wp(P(I,J)) = \wp(P(K,L)) \text{ and } C(P(I,J),P(K,L))) \qquad (1)$$

where C is the predicate

$$(C(X,Y) = T) \text{ IFF } (X \text{ IS CONNECTED TO } Y)$$

where "connected" means the existence of a sequence of points

$$(P(I,J) = S_1, S_2, \ldots, S_N = P(K,L)$$

where for each $J = 1,\ldots,N_J, S_J \sim P(I,J)$ and S_J, S_{J-1} are four neighbors.[8] This partitions P into connected homogeneous sets, which we shall call regions.

While other partitions could be used, this partition has the advantage that no information is lost, in the sense that the elements gathered together are not differentiable, and certain topological considerations, described below, are preserved in an easily manageable form.

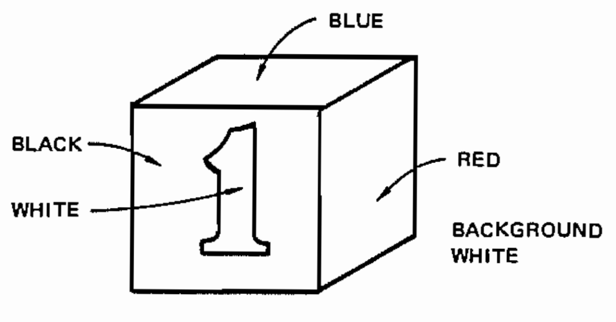
This first partition begins the formation of the data structure, forming the basic regions, called the elementary regions. The structure then encompasses the capability of conveniently joining regions in such a way as to form a description tree (as in Fig. 1) where each node is determined by a region, and each level is a new partition of the matrix.

II THE STRUCTURE AROS (A REGION-ORIENTED STRUCTURE)

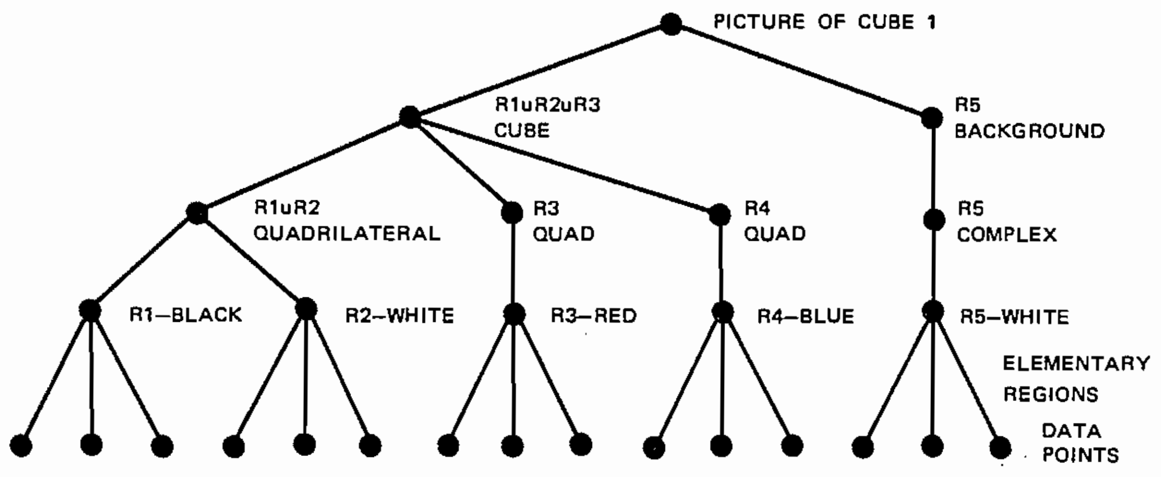
The matrix P = (P(I,J)) which provides the initial problem can be viewed as in Fig. 2; that is, each point P(I,J) is surrounded by four vectors--a counterclockwise-oriented boundary of the point. Formally then the boundary (denoted by ∂) of the point may be written as

$$\partial(P(I,J)) = V_1(P(I,J)) + V_2(P(I,J)) + V_3(P(I,J)) + V_4(P(I,J)) \quad ,$$

where the $V_1(P(I,J))$ are the vectors of Fig. 3.



(a) The Ideal Cube



(b) The Tree

TA-7494-8

FIGURE 1 A SIMPLIFIED TREE DESCRIPTION OF THE IDEAL CUBE SHOWN

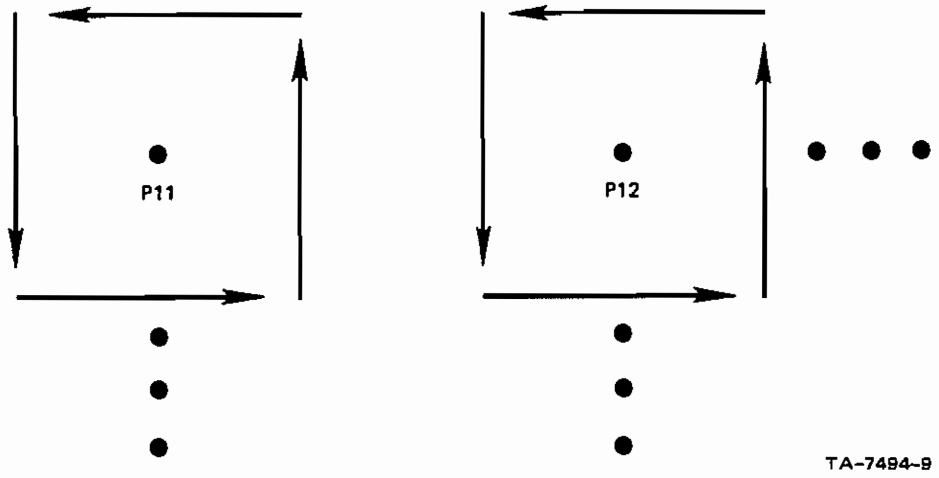


FIGURE 2 TWO POINTS OF A PICTURE MATRIX WITH THE SURROUNDING VECTORS SHOWN

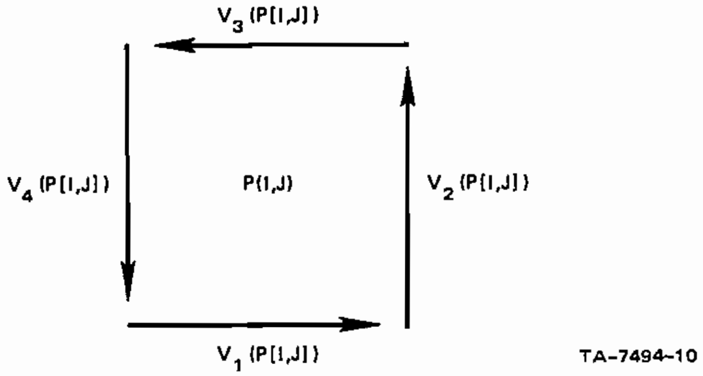


FIGURE 3 A MATRIX POINT WITH THE FOUR SURROUNDING VECTORS OF ITS BOUNDARY LABELED AS IN THE TEXT

Now, subject to the relations

$$V_1(P(I,J)) = -V_3(P(I + 1,J))$$

$$V_2(P(I,J)) = -V_4(P(I,J + 1)) \quad ,$$

it is apparent that the boundary of any set of points $X = \{P(I,J)\}$ is

$$\partial(X) = \sum_{P(I,J)\epsilon X} \partial(P(I,J))$$

and, in general, the boundary of any collection $\{X_n\}_{n=1}^{m}$ of disjoint sets is

$$\partial\left(\bigcup_{n=1}^{m} X_n\right) = \sum_{n=1}^{m} \partial(X_n) \quad .$$

In fact, this representation, though only instructive, is the formal background of what follows.

The algorithm (PARTITION) and description of Appendix A of this report give the method by which one can partition an arbitrary M x M matrix into elementary regions, each of which is represented by its boundary, as described above. The structure could be considered as a triple

$$S = (\text{DATA},\text{TOPOLOGY},\text{OPERATION}) \quad .$$

A. Data

Given the original data $\mathfrak{O}$ (the set of all the points of the a array considered as singletons), $\mathfrak{O}$ determines $\mathcal{U}$, the power set of $\mathfrak{O}$ (the set of all subsets of $\mathfrak{O}$). The domain of the structure data is the set $\mathfrak{C}$ of all connected elements of $\mathcal{U}$. The algorithm partition yields a subset $\mathfrak{H}$ of $\mathfrak{C}$, which is defined as the set of all the homogeneous elements of $\mathfrak{C}$ determined by the equivalence relation (1) (see Fig. 4). Furthermore, there is also a function $\mathcal{E}$ on $\mathfrak{O}$ onto $\mathfrak{H}$ which is defined by

$$\mathcal{E}:\mathfrak{O} \to \mathfrak{H}$$
$$o \to h$$

where h is the unique element of $\mathfrak{H}$ such that $o \subseteq h$.

The representation of each region in $\mathfrak{C}$ is by its boundary, which is a list of its components--the closed curves of the boundary (a region may be multiply-connected[8]). These curves are then, in turn, a list of vectors--ordered as they are geometrically (see Fig. 5).

B. Topology

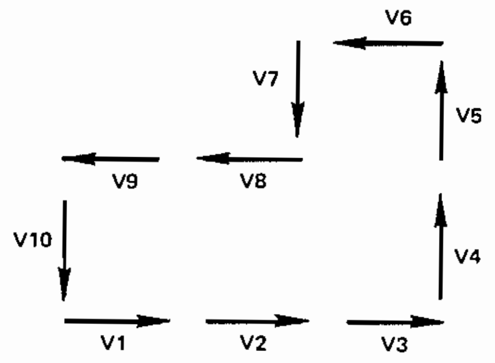
Implicit in this structure is a great deal of easily extractable topological (and indeed metric) information. In particular, we

$$\begin{tikzcd}
 & \mathcal{U} \arrow[d, dash] \\
 & \mathcal{C} \arrow[d, dash] \\
\mathcal{O} \arrow[ur, dash] \arrow[r, "\mathcal{E}"] & \mathcal{H}
\end{tikzcd}$$

TA-7494-11

FIGURE 4 THE DATA LATTICE SHOWING THE RELATIONSHIP OF $\mathcal{U}$, $\mathcal{C}$, $\mathcal{O}$, AND $\mathcal{H}$

(a)



(b) B = (C) = ((V1, V2, . . . Vn))

TA-7494-12

FIGURE 5 A TYPICAL BOUNDARY OF VECTORS (a) AND ITS CORRESPONDING LIST (b)

may consider the matrix P as embedded in E^2--the Euclidean plane. Thus, we can easily extract information such as what is to the left or right of a vector.

Thus, given the vector $V_K(P(I,J)) = ((X_T,Y_T),(X_H,Y_H))$, the point $P = (X_P,Y_P)$ to the right is given by the formula

$$X_P = (X_T + X_H \div (Y_H - Y_T))/2$$

$$Y_P = (T_T + Y_H \div (X_T - X_H))/2 \quad .$$

Knowing what is to the right or left of a vector makes finding the neighbors (i.e., regions whose boundaries are tangent to the boundary of a given region) trivial by virtue of the function $\mathcal{E}$.

Equally trivial is the question of whether a point is inside or outside a boundary, or any question as to the connectivity of the region (multiply-connected).

C. Operations

In order to be able to modify the structure as well as describe and retrieve information, an operation named MERGE was defined on disjoint pairs in C of Section A that joins the two regions, creates the new boundaries, and saves the local properties of each element. It may happen that we wish to cut a region into two by a curve (as determined perhaps by some higher level information), but it turns out that MERGE here will suffice once we define a negative boundary, which is just

$$-\partial(P(I,J) = (-V_1) + (-V_2) + (-V_3) + (-V_4)$$

$$-\partial(X) = \sum_{P\varepsilon X} -\partial(P) \quad .$$

Now we find X such that MERGE (-X,S),X are the desired regions. We note that, by definition, under the operation MERGE, the data is closed, context-independent, syntactically-descriptive, and flexible.

III THE PROBLEM DOMAIN

One can think of the problem domain as being the possible set of nodes for the description tree. Each node of the tree represents a region of $\mathcal{C}$, and associated with it is a summary of the description of the region it represents.

Trivially, one could consider all possible nodes of the tree subject to the constraints of the property function $\mathcal{P}$ and build them up as they make sense, but this is combinatorial. So, a more realistic way is to use proper heuristics to guide the path so that most merges are constructive, in the sense that the new nodes constructed at each step tend (in some sense) to be closer to the goal. We give an example in scene analysis.

The Stanford Research Institute robot lives in a limited environment in which the class of interesting scenes contain only cubes, wedges, and background. This class of scenes does not illustrate the more powerful aspects of the above structure, but it is the only example used so far. It will suffice as an illustration.

We let the description function $\mathcal{D}$ be

$$\mathcal{D}(P(I,J)) = \text{GRAYSCALE OF } P(I,J)$$

and we partition according to the above relation (1). The elementary regions are then regions of uniform grayscale.

Then, using the result of this pass and starting from one region, R_1, we grow the region by merging those neighbors that satisfy the following criterion:

If

R_n is a neighbor of R_1,

and if we put

$$I = \sum_{v \in \mathcal{J}} w_v v$$

where

$$w_k = \begin{cases} 1 & \text{if the gray scales of the point to the right and to the left of } v \text{ are different by } 1 \\ 0 & \text{otherwise} \end{cases}$$

and

$\mathcal{J}$ is the part of the boundary of R_1 tangent to R_n,

and if

P is the smallest of the perimeters of R_1 and R_n,

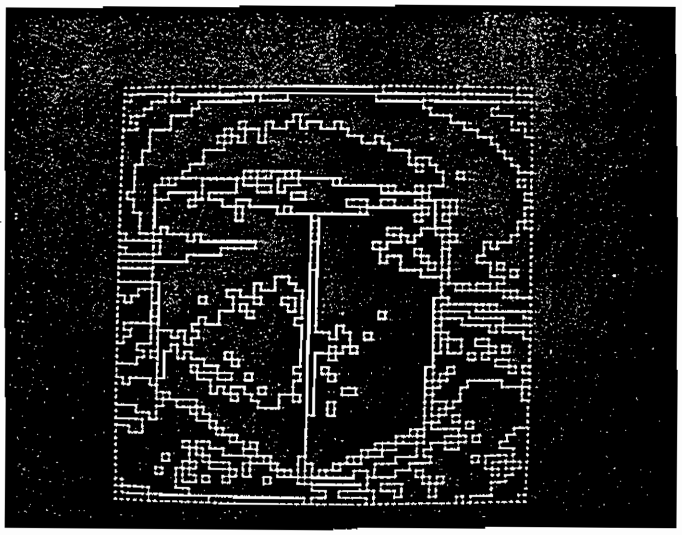
then the condition is satisfied if $I/P > \theta$, where θ is a threshold (we have used 0.5).

We continue to grow until this criterion is never satisfied for any neighbors of R_1. The whole picture or any portion of the picture can be processed in this fashion.

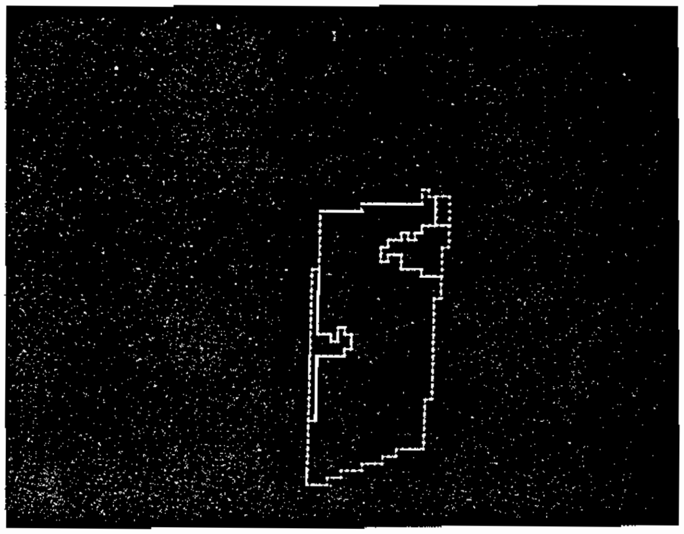
The result of this heuristic (see Fig. 6) is that we are ready to attempt a description of the scene.

IV CONCLUSION

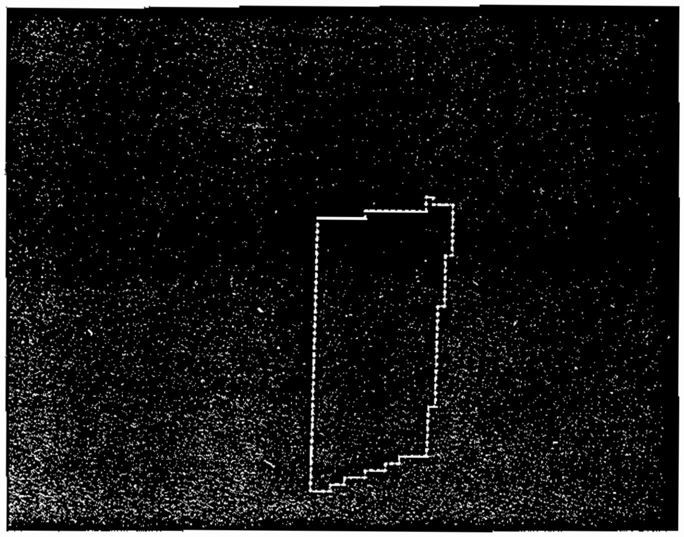
This type of representation does not pretend a great efficiency in coding or storing pictorial information. However, it is a structure that provides the ease of manipulation and topological richness necessary for picture-handling and description; it has the added advantage of being completely general.



(a) The picture partitioned into its elementary regions



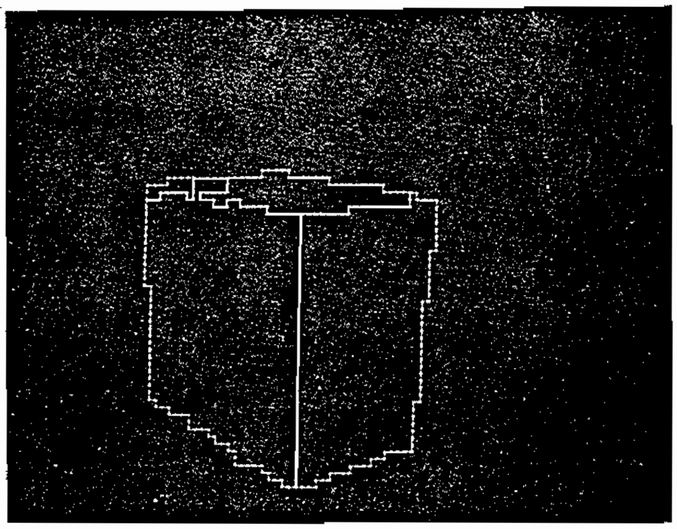
(b) One of the large regions of the picture together with the neighbors that satisfy the merging criterion



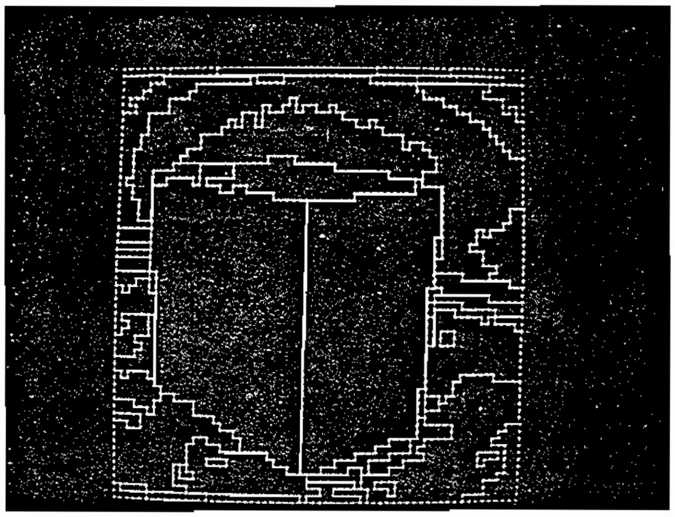
(c) Merge of all the regions that satisfy the merging condition (after several iterations)

TA-7494-13

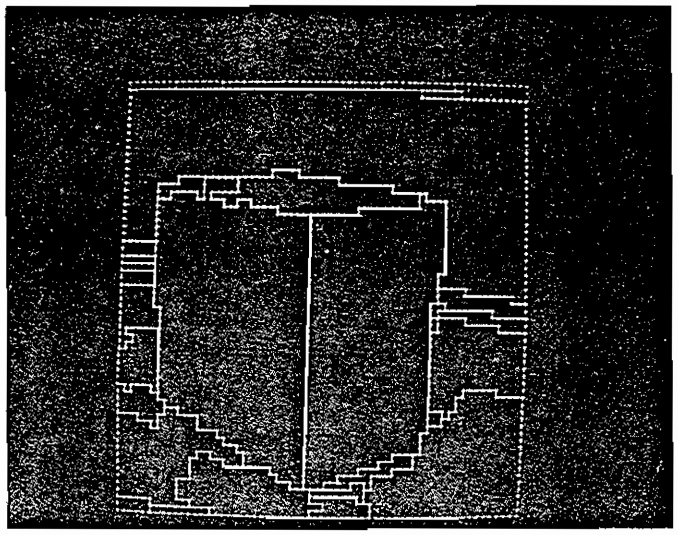
FIGURE 6 A TYPICAL PICTURE PROCESSED WITHIN THE STRUCTURE OF THE TEXT



(d) Four regions such as in (c) displayed together



(e) The regions of (d) in their background



(f) Picture completely processed in terms of the merging heuristic

FIGURE 6 A TYPICAL PICTURE PROCESSED WITHIN THE STRUCTURE OF THE TEXT Concluded

REFERENCES

1. I. E. Sutherland, "SKETCHPAD: A Man-Machine Graphical Communication System," Technical Report No. 296, MIT Lincoln Laboratory (1963).

2. B. G. Cook, "A Computer Representation of Plane Region Boundaries," The Australian Computer Journal (November 1967).

3. Pfaltz, Snively, and Rosenfeld, "Local and Global Picture Processing by Computer," Presented at Symposium on Automatic Photointerpretation, Washington, D.C. (May 1967).

4. R. G. Loomis, "Boundary Networks," Comm. ACM (January 1965).

5. H. Freeman, "On The Encoding of Arbitrary Geometric Configurations," IRE Trans. Elec. Comp. (June 1961).

6. C. T. Zahn, "Two-Dimensional Pattern Description and Recognition Via Curvature Points," SLAC Report No. 70, Stanford University (December 1966).

7. D. T. Ross, "A Generalized Technique for Symbol Manipulation and Numerical Calculation," Comm. ACM, Vol. 4 (1961).

8. A. Rosenfeld, "Picture Processing by Computer," Technical Report No. 68-71, University of Maryland (June 1968).

Appendix A

CONNECTED COMPONENTS ALGORITHM

1. Purpose of the Algorithm

The algorithm described below partitions an arbitrary $(M \times N)$ picture array into homogeneous connected components and represents each such region by the curves that make up its boundary.

2. Logical Description

The algorithm to find the connected components works on an array $(N \times N)$ of picture elements $P(I,J)$, each of which points to the list of its properties (intensity, color, texture,...). A two-dimensional Cartesian coordinate system is assumed where the elements of the picture stand in the positions defined by odd coordinates. The positions with even coordinates are the positions of the endpoints of the vectors used to separate picture elements (Fig. A-1).

Connectivity is defined by assuming that each point has four neighbors (Fig. A-1) and that two points--$P(I,J)$ and $P(K,L)$--are connected if there exists a sequence of points

$$P(I,J) = S_{M_1}, \ldots S_{M_N}, \ldots, S_{M_F} = P(K,L)$$

such that $S_{M_{N-1}} \sim S_{M_N}$ and $S_{M_{N-1}}$ and S_{M_N} are neighbors. ($\sim$ is for the equivalence relation between the elements of the picture.)

Given the equivalence relation $\sim$, this definition of connectivity, and an array of picture elements, the algorithm finds the connected components homogeneous with respect to $\sim$ (regions) and their boundaries (the closed curves bounding the region).

Each picture element of the original array is replaced by the name of the region to which it belongs. The name of the region points to the list of the common properties of this region and to the boundary of that region.

In general, a boundary separates two or more regions; parts of the boundary or even the whole boundary of a given region are also parts of the boundary of another region. A simple representation is used here that avoids the orientation problem or multiply-branching nodes is to create for each region its own boundary, so that two contiguous regions do not share parts of the same boundary (Fig. A-1).

The boundary is computed as an ordered list of vectors of unit length such that, for each vector of the boundary of a given region, the region is on the left side of this vector, and such that any vector of the list is followed in the list by the vector that follows it geometrically. Any vector is connected at least to one and at most to two other vectors.

The algorithm itself applies the L-shaped window of Fig. A-2 to each point of the picture, scanning in a raster and performing what is best described by the flow chart of Fig. A-3.

To process the edges corresponding to the frame of the picture, the picture is expanded by one element on all sides to a picture of $(N + 2) \times (N + 2)$ elements, by adding a region with such values of the properties that the equivalence is never true with any element of the original picture.

Since any path between two points of the picture satisfying the definition of connectivity could be generated by this type of window applied to every point of the picture, it is obvious that this algorithm will give the connected components of the picture. However, it is possible that one region could be first recognized as two or more different regions at the beginning of the process (see Fig. A-4); they will be joined further on under the same name. The partial results computed (boundaries) will be also joined.

A vector created by the window is added to the list of vectors representing a component of the boundary (since a boundary could have more than one component if there are holes in the region) if and only if one of the endpoints of this vector is the same as one of the free endpoints of the list; otherwise, this vector begins a new component. Then, when a component has been changed by adding a vector or a list of vectors, the connection of this component with the other components of the boundary

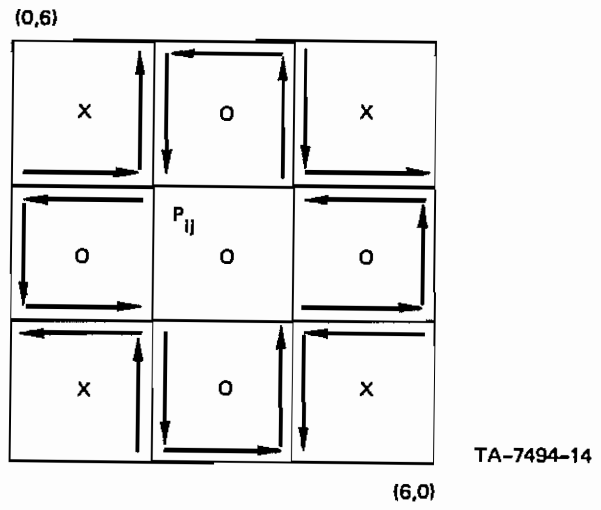


FIGURE A-1 A SAMPLE OF A PICTURE ARRAY WITH THE BOUNDARY VECTORS SURROUNDING A CONNECTED COMPONENT, NAMELY THE O's

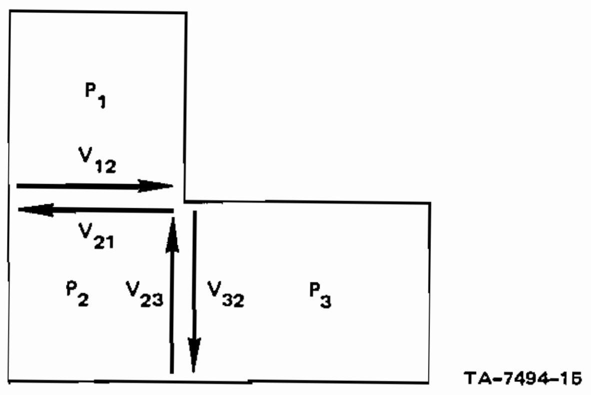


FIGURE A-2 THE WINDOW FOR THE PARTITION ALGORITHM

is tried. It is possible that two originally separated components will be joined later (Fig. A-4).

3. Pictures

This algorithm was developed for a scene-analysis purpose, to describe the picture in a "region-oriented sense" and to provide a structure flexible enough to be able to modify regions by joining them together. The algorithm was tested on grayscale pictures of different sizes (from 120 × 120 to 60 × 60) of geometrical objects, with 16 levels of gray. The equivalence relation was that the grayscale of two contiguous elements should be equal.

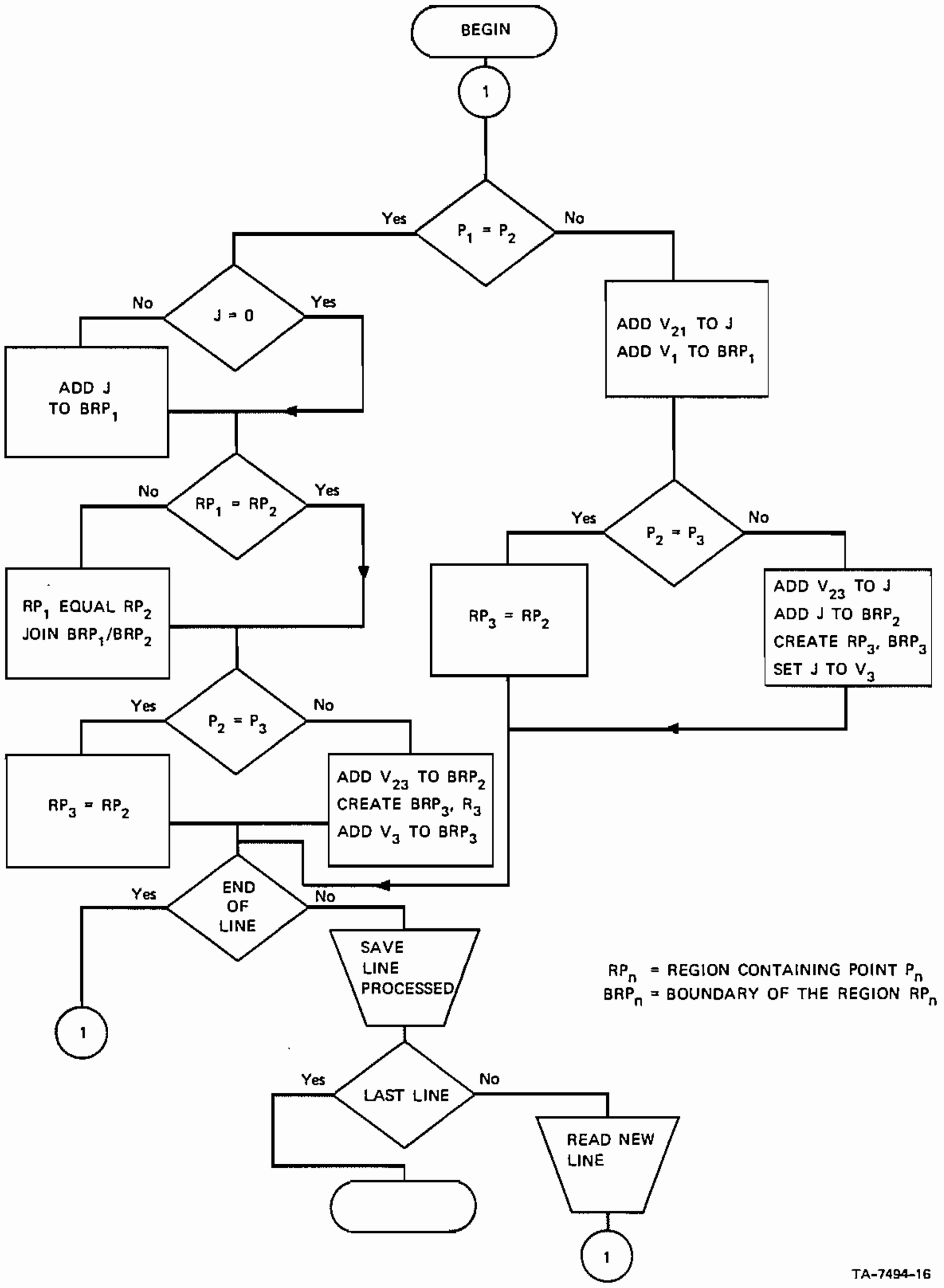


FIGURE A-3 FLOW CHART OF THE CONNECTED COMPONENTS ALGORITHM

It is planned to use it in another kind of environment for pictures, taking into account different properties like color, range of each point for stereo pictures, and a kind of texture measurement attached to every point.

4. Implementation

At present, this algorithm is implemented at SRI in LISP 1.5 on the Artificial Intelligence Group's SDS-940 as a study in picture-processing. LISP was chosen for this study because it was the most natural available language in which to do this type of processing, since there are numerous lists constructed that by nature are of indefinite length, and the structure of the language itself lends itself to the type of structure we have described.

In this implementation we represent the original picture as an array, each element of which is a pointer to a list of properties (a property list). The function partition then has this array as data and returns the above structure. The array elements now point to the name of the elementary region to which they belong (an atom created for this purpose). This atom then has on its property list the properties common to all the points of that region, and the value of that atom is the boundary of the region. This boundary is a list of its components, which are in turn lists of the vectors of which they are composed.

The functions D and E are made up of GETP and SETA respectively (see Fig. A-5).

The disadvantage to LISP on the SDS-940 is that it is very slow. This inefficiency could be reduced by the use of some special language, or, perhaps better still, by LISP on a LISP-oriented rather than a FORTRAN-oriented machine (that is, one with a large core and a proper set of machine instructions).

One plan to speed up computation (whatever the machine) is to use a zoom technique. One takes an N X N picture and reduces the resolution to, say, 1/4 N or even more, and processes the course picture first to find the section of interest. Once the desired information is extracted, ambiguities

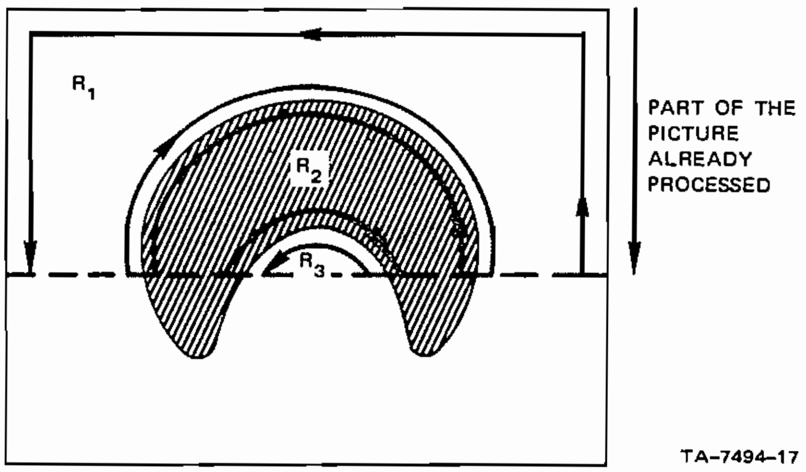


FIGURE A-4 A PARTIALLY PROCESSED PICTURE (R_1 and R_3 are first thought of as separate regions. It is not until all of R_2 is processed that R_1 and R_3 are joined.)

$$\begin{tikzcd}
\text{ARRAY (I)} \arrow[r] & \text{RK} \arrow[r] \arrow[dr] & \text{BRK} \arrow[r] & \text{((V1, . . . , Vn) . . .)} \\
 & & \text{(COLOR RED INTENSITY 5 . . .)} &
\end{tikzcd}$$

TA-7494-18

FIGURE A-5 A DIAGRAM OF THE IMPLEMENTED STRUCTURE IN LISP AT SRI. The function $\mathcal{D}$ is composed from the LISP function GETP, and $\mathcal{E}$ is essentially EVAL.

are resolved as much as possible by increasing the resolution locally. It has been found that often the figures of a picture can be determined in a much reduced resolution and that the high resolution is needed only locally.