MASSACHUSETTS INSTITUTE OF TECHNOLOGY

PROJECT MAC

Artificial Intelligence
Memo. No. 177

August 1969

Preprocessor for Programs which Recognize Scenes

H. N. Mahabala*

A visual scene is transformed from a very simple and convenient format, to an internal format which describes the same scene, but is more akin to complex manipulations. This format is compatible with programs like "SEE".
The entire analysis is done using a basic primitive which gives the orientation of a point with respect to a directed line.
A novel handling of inaccuracies in the scene is achieved by considering the lines to be strips of small but not negligible width. The criterion is very general and easy to modify.

* Visiting Professor from Indian Institute of Technology, Kanpur, India.

Preprocessor For Programs Which Recognize Scenes

H. N. Mahabala
(Visiting Professor from Indian Institute of Technology, Kanpur, India)

The output of a program which processes the data from a vidisector is often a graph (list of vertices, their location and list of neighbors). This graph has to be checked for consistancy such as all intersection of links are labelled as vertices, etc.. Further the regions have to be labeled and property lists which organize the ingormation in very specialized forms for programs like "SEE" have to be prepared. A preprocessor called "SETUP" has been written in basic LISP. SETUP includes the following features.

(a) Entire analysis is performed using a basic primitive which gives the orientation of a point with respect to a directed line. Orientation is expressed as to the left, or to the right or in the direction of the line or away from the directed line. Since inaccuracies in the location of a vertex is unavoidable the directed line is in fact interpreted as a strip of variable width. Criterion used is very general but the organization is such that only one function has to be changed if a new criterion is to be used.

(b) Since advice from lower level programs deal with only points (for example: points X and Y are on the same body) appropriate region names can be determined using available functions.

(c) Main background is determined without additional information.

It is interesting that the following can be achieved using only the "orient" primitive.

(i) Determine whether a given point is inside or outside a general region (with holes).

(ii) Determine whether a given boundary encloses a region with bounded or unbounded area.

(iii) Determine whether a given line segment intersects the boundary of a region.

(iv) Determine the connectivity of a graph and relative location of the various subgraphs.

1. Introduction

Programs which recognize or understand scenes work with line diagrams of a scene. It is preferable for the data of the scene to be presented in a pseudo canonical form. This requires ordering of the neighbors of a vertex (for example, anticlockwise) and including the names of regions in describing the neighborhood of a vertex. Since one has to test a program (in turn a heuristic) for a large variety of scenes of considerable complexity preparation of data in the pseudo canonical form can be very time consuming (see Figure 'HARD'). So it is desirable to have a preprocessor which can accept the scene in a simple format such as the incidence matrix. Further the output of programs that process the output of a vidisector is often in the simplified format. Thus the preprocessor can act as a link between the lower level programs and higher level programs. It is very desirable, why even necessary, for work on recognition programs to be able to use as data a line diagram (in black on white) of a scene to be analyzed. Thus one can test a program on a wide variety of scenes without too much trouble and facilitate answering the most often asked question of a recognition program: "What will the program do on this scene?".

Instead of treating the preprocessor as merely a program (or hack) it would be nice if recognition primitives and computational techniques involved are carefully selected so that one can view the preprocessing as the computation of an automaton with certain capabilities. Such a view point can help in arriving at a hierarchy of automaton for understanding scenes.

2. Objectives

2.1 General

(a) The output of the program should be compatible with input for existing recognition programs. However better organization of output wherever desirable should also be considered.

(b) Program should be in LISP and LISP features used should be general enough to run on almost any version of LISP. Compilation if desired should also be straight forward.

(c) All functions defined should be prefixed by a special character to facilitate easy change of a name (M% is used) in case of conflict that might arise when the program is made part of another program.

(d) Remprop all unnecessary functions and variables at the end of the preprocessing.

(e) The output will usually be in the form of property lists. Hence it should be possible to print out the output of the program in a format easy for human understanding.

(f) All error processing should be centralized in a single routine (M% ERROR) so that handling of an error can be altered very easily. This requirement is mandatory if the whole recognition system should somehow procede in case of certain errors and not just hangup. (At present the preprocessor quits after printing a suitable message.)

2.2 Particulars

(a) The input of the graph of a scene should be as simple as possible. The data includes name of scene, and list of vertices, their location and unordered list of neighbors.

(b) The output is in the form of property lists and a subset of the property lists is compatible with "SEE" of Guzmán.

(c) It is sometimes necessary to handle more than one scene with perhaps same vertex names at the same time. This is made possible by including a feature by which names of vertices, regions and subgraphs can be prefixed by the name of the scene. All global properties are on the name of the scene.

(d) Check for certain types of constancies in the input such as:

(i) Each link of the graph appears in two places, i.e. if B appears as a neighbor of A then A should appear as neighbor of B. The urge to relax this requirement (since finally a lower level program prepares input) and update suitably in case of error was curbed so that human errors in preparation of data is not misinterpreted.

(ii) Intersection or meeting place of any two links should be at a vertex.

(iii) Two or more neighbors of a vertex should not be collinear on the same side. (See also section 4.2).

(e) Since input data and in particular location of a vertex is subject to hardware inaccuracies, not to forget truncation errors, certain decisions such as the following should not be based on pure geometric theorems. (See also section 3.1.3).

(i) Whether two links are in the same line.

(ii) Whether three points are collinear.

(f) Find the connectivity of the input graph. Separate the scene into subgraphs and find their relative location. This is desirable since each subgraph can be a body or a hole.

(g) Find the main background (the only unbounded region which extends to infinity around the graph). The background of each subgraph (the region in which the subgraph is situated) should also be determined. The information on each subgraph should be organized in such a way that recursive analysis of the scene can be facilitated. Matching T's cannot be found satisfactorily with the available information, however, matching T's with respect to the main background should be determined. There should be facility for updating matching T's as and when more is known about the scene.

(h) It should be possible to change the name of a vertex, region or subgraph in a scene even after the preprocessing is complete. (Such as identifying vertex A on LEFT scene is same as vertex X in RIGHT scene of a stereo pair.)

(i) Advice from lower level programs or between higher level programs such as

(i) Point X (XCOR, YCOR) is in the background.

(ii) Points X and Y are on the same body.

(iii) Region which has KV* (ordered boundary) *is* . . . .

(iv) Region with A and B (two ordered adjacent points on the boundary) on the boundary is...........

should be updated by substituting internally generated names (for subgraphs or regions). This is facilitated by making available some utility functions which are retained even after unnecessary ones are removed.

3. Interesting Features

3.1 "Orient" Primitive

In conforming with the intention to use simple recognition primitives it was felt that all the work involved in the preprocessor, referred to as SETUP, can be done by using a primitive which answers the question: "What is the orientation of a point with respect to an oriented half line." The answer will be in the form: to the left, to the right, ahead or behind. This recognition ability is more basic than evaluating magnitude of angles. Further one always makes approximations in judging orientation and hence similar facility for approximating should be incorporated in this primitive. This approximation is all the more necessary in the computer program since location of vertices are subject to hardware errors. Truncation aggravates the problem.

3.1.1 Strict Mathematical Rule Such as:

0° - ahead

$0^\circ < \theta < 180^\circ$ left

180° behind

$180^\circ < \theta < 360^\circ$ right

where θ is the angle between the line and the line joining the starting point of the oriented line and the given point (angle (AB) in Figure 1).

This criterion is rejected because there is no room for approximation and if this criterion is used no T's will be found in real scenes!

3.1.2. This incorporates an approximation to criterion 3.1.1.

$-10^\circ \leq \theta \leq 10^\circ$ ahead

$10^\circ < \theta < 170^\circ$ left

$170^\circ \leq \theta \leq 190^\circ$ behind

$190^\circ < \theta < 350$ right

this works fine except that very large error results if the point distance from the origin of the directed line is large. In general one can expect that all vertices shift by about the same amount irrespective of their location due to hardware inaccuracies, (barring edge effects). Hence moving a vertex by large distance in order to consider it collinear with a line is not in general justified.

3.1.3. The line is viewed as a strip of finite width for deciding collinearity. This in itself will not suffice because orientation of C with respect to $\vec{AB}$ may not be collinear whereas A is collinear with respect to $\vec{BC}$ (See Figure 1(b)). Even though successive logical extensions cannot be allowed in approximate techniques at least three points should be judged collinear no matter which two of the three points is used for the directed line. So three points are judged collinear if any one point is collinear with respect to the strip along the other two points, i.e. A, B and C are considered if:

C lies in a strip of width ±EPS along AB

or B lies in a strip of width ±EPS along AC

or A lies in a strip of width ±EPS along BC

It looks at the outset that taking the strip parallel to one of the three sides of the triangle is not general enough. In fact one would like to fit a line which minimizes the sum of the squares of the distance from each point to the line. And the three points are collinear if they lie within a strip of width ±EPS about this line.

It is not easy to find such a line.(This is not the regular least square fit line). Further, it can be easily shown that, if three points are collinear from the best fit line criterion, it will also be collinear with respect to our three strip criterion.

Figure 2(a) shows the criterion used in the present preprocessor (M%SETUP). The organization of the preprocessor is such that one easily can use any other criterion by rewriting M%ORNT. Figure 2(b) illustrates the case where EPS is larger than distance between A and B. Obviously greater reliance is put in direction of AC or BC than that of AB.

It is reasonable to use a value for the half width of the strip (EPS) which is commensurate with the hardware inaccuracies. Increasing EPS beyond reason (half of minimum distance between any two neighbors) will cause trouble which is understandable. All attempts to adjust value of EPS dynamically by studying only the neighborhood of a vertex should be considered carefully. One of the dangers of such an adjustment is that many vertices will end up being T's!

One interesting technique of analysis of a scene would be analyze it at more than one value of EPS and reconcile the discrepancy if any.

It should be noted that M%ORNT only shows conclusions about the orientation and does not adjust the coordinates of the vertices to fit the orientation geometrically (according to 3.1.1). It is feared that any attempt to adjust data recursively will ruin the features.

3.2 "SECT" Function

A primitive derived from "orient" finds whether two line segments intersect or not (Figure 3).

3.3 "Inside" Function

It is interesting that we can determine whether a point is internal to a region (given by ordered vertices on the boundary) by using the orient primitive. A point is inside if the number of intersections of the boundary with the infinite half line joining the point to a vertex on the boundary is odd, otherwise outside.

3.4 "Bounded" Function

Often we need to find if a region given by an ordered set of vertices has bounded area. The technique first finds the vertex with the minimum X coordinate. Then if the region is to the left of this point it is unbounded (Figure 2).

4. Overall Technique (M%SETUPX)

The names referred to in this memo (prefixed by M%) are names of LISP functions.

4.1 M%SETUP1

Input data is separated after appending a prefix (NIL if M%SETUP is used and NAME if M%SETUP* is used).

4.2 M%ARRANGE

The neighbors of each vertex are arranged in counter clockwise order. This is achieved by successively adding a neighbor to the ordered list. If EPS2 (square of the half width EPS for M%ORNT) is large two vertices will fall in the same direction which is an error.

4.3 M%TYPE

The TYPE* (see section 5) of each vertex is found.

4.4 M%FREGION

Regions associated with each boundary is labeled. The technique is: start with a vertex A (Figure TEST) and choose a neighbor (say H) connected to a A by an unused link (i.e. $\overrightarrow{AH}$ has not yet been identified as a boundary of a region). Mark $\overrightarrow{AH}$ as used. At H pick the vertex (K1) which is before A in the ordered list of neighbors of H. $\overrightarrow{HK1}$ should also be unused which will not be the case if data is not consistent. So we have AHK1 as a partial boundary of a region :4. Proceed as before from K1 until we reach A. There make sure that the next link (after $\overrightarrow{MA}$) is indeed $\overrightarrow{AH}$. This will not be the case if we had started from H instead of A. So proceed until the first link used is obtained again. It should be noted that a vertex can appear more than once on its boundary (:4 has as K vertices H,K1,L1,J,I,H,G,F,E,D,C,B,M,A). At this stage the region names and their boundaries are not final for those regions which have holes. For example: entire area within A2 A7 A6 A3 (Figure CRAZY) will be :3 and entire area outside D1 D4 D3 D2 will have a separate number. Later the extra name is eliminated and the boundary of:3 is adjusted.

4.5 M%SUBGRAPH

Connectivity of the total graph is found. The technique is that each subgraph is constituted by a minimal subset of vertices whose neighbors are also in the subset. For example, G1, G2 and G3 (Figure CRAZY) form a subgraph.

4.6 M%SEPARATE

The relative location (structure) of the various subgraphs are determined. The technique is: the background (unbounded region associated with a subgraph) of a subgraph is determined (Section 3.4).

Pick the subgraph which has the vertex with the minimum X coordinate of all the vertices (A1 of SUBGRAF10 in Figure CRAZY). The background of the scene is the background of Subgraf 10 (:2). Separate the other subgraphs according as OUTSG if one point on them is in :2 (i.e. outside SUBGRAG10 and INSG if one point in them is not in :2. For example subgraphs 7, 4 and 6 will be in OUTSG and 3.2.5 and 1 in INSGμ Separate INSG into subsets within each region of SUBGRAF10. Proceed with OUTSG and various subsets of INSG from beginning (recursively) except that the background in each case is to be merged with a region of SUBGRAF10. For example background of OUTSG (associated with C1 C2 C3) will have to be merged with :2 and background of 1,2 and 5 (associated with F1 F2 F3 F4) is to be merged with :1. The structure information is organized as a list and is explained in Figure CRAZY.

A note about finding the background (M%BGROUND). Any cardinal point (i.e. a vertex with minimum x or y coordinates or with maximum X or Y coordinates) can be used for finding the background. The region name in each case should be the same. However, if the intersection of two links is not identified as a vertex the region names will be different causing an error exit.

4.7 M%ADJUST1

Merging information of multiple region names are gathered and information about the subgraphs in each region is collected.

4.8 M%REDUND

If there is more than one name possible for a region one with a lower number is retained. The property list of region names are updated (take care of multiple boundaries) and property lists of unused region names are deleted. However, the occurrences of region names eliminated have yet to be replaced by corresponding names. A list (M LIST) of substitutions necessary is prepared.

4.9 M%REPLACE

M LIST is used to make the substitutions in all property lists associated with M%SETUP (or M%SETUP*).

4.10 M%SEEDATA

Since one main requirement of M%SETUP was to be compatible with "SEE" appropriate new properties are created. NAME is given a value which is compatible for using "PREPARA" (a preprocessor for SEE). However, PREPARA can be bypassed. Value of NAME can be used to plot the scene using a package written by Guzmán.

4.11 M%MATCHT

A pair of T's in a scene are matching T's if

(i) they face eachother with their tails collinear and are the closest possible,

(ii) the base region (one which subtends an angle of 180° at the vertex) of each T should not be a background, and

(iii)the line joining them does not intersect any region identified as background.

M%SETUP finds the matching T's only with respect to the main background (i.e. 35 in Figure HARD). However, M%MATCHT is so organized that further updating can be done very easily. For example, M%MATCHT can be called using 36 and 34 as background and operating on only those T's which already have mates. In this case it well not make an alteration in the set of matching T's. M%MATCHT was tested on HARD by identyfying 26 (rather illogical) as background. Then vertices a, fa and b, fb were eliminated from the set of matching T's.

The technique used is to make sure that each T(of a matching pair) accepts the other as a mate, when all T's are available for matching. Since M%ORNT uses a strip for matching collinearily sometimes, cycle can arise which is an error.

5. <u>Property Lists</u>

M%SETUP (or M%SETUP*) sets up a number of property lists whose description follows.

5.1 NAME

Its value is data suitable for PREPARA.

PREFIX	Explode list of characters in NAME if M%SETUP* is used, otherwise NIL.
ACCURACY	Half width of the strip for M%ORNT.
ERROR	Error message in case M%MATCHT had to delete vertices from matching because of cyclicity. [Such as B is mate of A, C is mate of B, and A is mate of C.]
VERTICES	List of vertices in the scene.
REGIONS	List of regions in the scene.
CONNECTIVITY	Number of subgraphs in the scene.
SUBGRAPHS	List of subgraphs in the scene.
MAXMIN	List of cardinal points (Min X, Max X, Min Y, Max Y).
BACKGROUND	List of backgrounds (only one initially).
STRUCTURE	Structural information as explained in Figure CRAZY.

5.2 Nth Subgraph (SGRAFN)

MAXMIN VERTICES REGIONS BACKGROUND	Same as in case of Name (5.1).

5.3 VERTEX

XCOR	X-Coordinate
YCOR	Y-Coordinate
TYPE	Type information. See references to GUZMAN for details.
KIND	List of counter-clockwise neighboring vertices and regions (alternately).
N VERTICES	List of neighboring vertices.
N REGIONS	List of regions having VERTEX on the boundary.
NEXTE	Name of matching T, otherwise NIL.

COOR	List of XCOR and YCOR.
TYPE*	Classification of the vertex. Types L, ARROW, PEAK, X, T, MULTI, K, FORK are the same as GUZMAN new types.
KK	One straight line and more than two branches on one side (vertex G in Figure TEST).
SL	Straight line (I in TEST).
CROSS	Two straight lines cross at the vertex (B in TEST).

KK becomes MULTI, CROSS becomes X in TYPE. However, SL continues to be SL which may cause error in SEE. However, SEE as it stands does not expect SL type of vertices.

OVERTICES	List of elements each of which is a list of a neighbor and an orientation. Orientation gives a measure of the angle to be rotated (anticlockwise) from the neighbor to reach the next neighbor. For example at A in TEST $M\hat{A}B<180^\circ$ is denoted L, $H\hat{A}M>180^\circ$ (R) and at G $F\hat{G}H=180^\circ$ (CA).

5.4 REGION

FOOP	See GUZMAN
NEIGHBORS	List of adjoining regions.
KVERTICES	List of all vertices on the boundary. No separation even if there are multiple boundaries.
BACKGROUND	Has 'BACKGROUND' if region is a background, otherwise NIL (:4 in TEST).
BOUNDARY	List of elements. Each element is a list of links in one branch of the boundary. Each link is a list of ordered (counter-clockwise) neighbors on a segment of the boundary (:1 and :2 in TEST).
OTRAVERSE	List of elements. Each element is a list of subelements in one branch of the boundary. Each subelement is a list of a vertex on the boundary and a measure of the angle subtended by the region at that vertex.
NEIGHBORS*	List of elements. Each element is the ordered NEIGHBORS of one branch of the boundary.
KVERTICES*	List of elements. Each element is the ordered KVERTICES of one branch of the boundary.
INSUBGRAPH	List of subgraphs located in the region.

can be done once the two OTRAVERSE'S are compatible. OVERTICES can also be used similarly to vertices.

(viii) Choose accuracy very carefully.

(ix) Since each subgraph can be a hole or a body, separate analysis of a subgraph is facilitated by the properties on each subgraph.

(x) STRUCTURE can be used to match scenes at a gross level.

REFERENCES

1. Guzman, A. Decomposition of a visual scene into three-dimensional bodies. Prodeedings of the AFIPS Fall Joint Computer Conference, Vol. 33, Part One, pp 291-304, December 1968. Review 17,227; Computer Reviews 10, 8. August 1969. Also available as Project MAC Memorandum MAC-M-391, A.I. Memo. No. 171.

2. Guzman, A. Computer Recognition of three-dimensional objects in a visual scene. Ph.D. Thesis, Electrical Engineering Dept. M.I.T. December 1968. Also available as a Project MAC Technical Report MAC-TR-59.

6. Use of M%SETUP

(i) Use M%SETUP* if more than one scene with same vertex names have to be handled simultaneously.

(ii) Even if very little information is changed in the scene it will have to be set up again.

(iii)The property lists created by M%SETUP can be printed out by using M%ANSWER.

(iv) The property lists can be stored on disk on disk in a suitable internal format so that the property lists can be reestablished by reading the file. Consult GUZMAN for the program.

(v) Since M%SETUP finds matching T's only with respect to the main background updating is necessary every time background information is altered. This should be all right since one may not analyze scenes with holes in the main background. Hence in general matching T's will be a subset of the matching T's found by M%SETUP.

(vi) Advice from the lower level programs can be updated by using the special functions available (See SETUP Comment).

(vii)Matching models of regions is facilitated by the property OTRAVERSE. Exact comparison of lengths of sides and magnitude of angles

BRIEF MANUAL FOR USING THE FILE VISION SETUP.

THE PACKAGE PREPARES A DATA GIVEN IN THE FORM

```
(SETQ DATA (QUOTE (NAME
          (VERTEX1 (X1 Y1) ( NEIGHBORS1))
          (VERTEX2


          (VERTEXN                 )))))
```

IN AFORM COMPATIBLE WITH "SEE". ADDITIONAL PROPERTIES ARE ALSO INCLUDED.

EXECUTE (MSSETUP DATA ACURRACY)

THE VALUE OF ACURRACY IS HALF THE WIDTH OF A STRIP WITHIN WHICH THREE POINTS SHOULD LIE. FOR BEING CONSIDERED COLLINEAR.
IF ACURRACY IS ZERO GEOMETRICAL COLLINEARITY WILL BE USED.

THE PROGRAM TERMINATES WITH "DONE".

THE PROPERTY LISTS ON NAME, VERTICES, REGIONS AND SUBGRAPHS CAN BE PRINTED OUT BY

1. LOAD THE FILE VISION ANSWER
2. EXECUTE (MSANSWER SCENE NIL).

GLOBAL VARIABLES USED ARE
SCENE EPS2 PREFIX RGPREFIX SGPREFIX

THE VALUE OF SCENE IS "NAME".

VALUE OF "NAME" IS DATA IN A FORMAT COMPATIBLE WITH THE INPUT REQUIRED BY PREPARA OF "SEE".

ALL FUNCTIONS ARE EXPRS AND THERE NAME BEGINS WITH MS.

THE REGIONS ARE NUMBERED CONSECUTIVELY FROM #1 WHEN A NAME IS USED FOR THE FIRST TIME AFTER LOADING A LISP.
NUMBERING CONTINUES EVEN IF THE DATA IS RESETUP.
HOWEVER THE NUMBERING CAN BE STARTED FROM ANY VALUE BY RESETTING THE PROPERTY RNAME ON NAME TO STARTING NUMBER REQUIRED.

CAUTION IF THE VALUE OF ACCURACY IS TOO BIG ERROR WILL OCCUR IN REARRANGE.
MAXIMUM VALUE FOR ACURRACY IS ABOUT ONE HALF OF OF THE MINIMUM DISTANCE BETWEEN ANY NEIGHBORS.

"NAME" CAN BE APPENDED TO NAMES OF VERTICES REGIONS AND SUBGRAPHS BY EXECUTING
(MSSETUP* DATA ACURRACY)
PROGRM TERMINATES WITH "DONE*".
EXECUTING (MSANSWER SCENE PREFIX) PRINTS OUT THE PROPRTY LISTS WITHOUT THE PREFIX.
EXECUTING (MSANSWER SCENE NIL) PRINTS THE INTERNAL FORMAT (WITH PREFIX).

THE EXECUTION OF MSSETUP OR MSSETUP* AUTOMATICALLY REMPROPS ALL FUNCTIONS EXCEPT A FEW UTILITY PROGRAMS
MSSETUP CAN BE USED BY READING IN THE FILE VISION SETUP AGAIN.
IF YOU DONT LIKE THIS REMOVE THE LAST S-EXPRESSION IN MSSETUPX.

IF NO FUNCTION IS TO BE RETAINED EXECUTE (MSA_CLEAR)

BRIEF DESCRIPTION OF FUNCTIONS RECOMMENDED FOR USE

```
NOTE. KV* IS THE ANTICLOCKWISE TRAVERSE OF THE BOUNDARY OF AREGION AND IS ALIST OF LISTS.
FOR EG KV* OF A TRIANGULAR REGION IS ((A B C)) WHERE A B & C AR VERTICES.
IF THERE IS AN INNER TRIANGLE X Y Z THEN KV* OF THE ANNULAR REGION IS ((A B C) (F E D)).
NOTE THAT NO VERTEX IS REPEATED.
        DOUBLE LETTER (SAY AA) MEANS THE VARIABLE IS ALIST OF XCOR AND YCOR (OF A).
        FUNCTIONS MSBOUNDED  MSINSIDE* AND MSRSECT* REQUIRE THE PROPERTY
 COOR (LIST OF XCOR YCOR) ON THE VERTICES MENTIONED IN KV*

(MSORNT EPS2 AA BB CC)

        RETURNS THE ORIENTATION OF CC WITH RESPECT TO THE DIRECTED LINE AA BB.
THE THREE POINTS ARE COLLINEAR IF THEY ALL LIE WITHIN A STRIP OF WIDTH 2*(SQRT EPS2).
THIS CRITERION IS SAME AS FITTING THE LEAST SQUARE FIT LINE TO THE POINTS AND MAKING SURE
 THAT DISTANCE FROM ANY POINT TO THE LINE IS LESS THAN (SQRT EPS2).

IF CC IS COLLINEARRETURNED VALUE IS
        CT  IF CC IS IN SAME DIRECTION AS BB (WITH RESPCT TO AA)
        CA  IF CC IS IN A DIRECTION OPPOSITE TO THAT OF BB
        CO  IF CC IS WITHIN EPS2 OF AA

IF CC IS NOT COLLINEAR WITH AA BB THEN
        L  IF CC IS IN THE LEFT HALF PLANE OF THE DIRECTED LINE AA BB
        R  IF CC IS IN THE RIGHT HALF PLANE OF THE DIRECTED LINE AA BB


(MSDIST2 AA BB)
        RETURNS THE SQUARE OF THE DISTANCE BETWEEN AA AND BB


(MSSECT AA BB CC DD)
        FINDS IF THELINE SEGMENTS AA BB AND CC DD INTERSECT AND RETURNS
        ON  IF POINT OF INTERSECTION IS ONE OF AA BB CC DD
            (COINCIDENCE MEANS WITHIN ADISTANCE OF (SQRT EPS2))
        YES IF POINT OF INTERSECTION IS OTHER THAN AA BB CC DD
        NO  IF THEY DO NOT INTERSECT


(MSBOUNDED* KV*)
                FINDS IF THE REGION GIVEN BY KV*HAS FINITE OR INFINITE AREA AND RETURNS
        BOUNDED   FINITE
        UNBOUNDED  INFINITE



(MSINSIDE* KV* XA)
        FINDS IF THE POINT XX IS INSIDE (INCLUDING ON) OR OUTSIDERETURNS
        IN
        OUT
*


(MSRSECT* KV* XX YY)
        FINDS IF THE LINE SEGMENT XX YY CUTS THE BOUNDARY OF REGION GIVEN BY KV* AND RETURNS
        ON  IF POINT OF INTERSECTION IS XX OR YY
        YES  IF POINT OF INTERSECTION IS OTHER THAN XX OR YY
        NO  IF THERE IS NO INTERSECTION


(MSREGIONPT SCENE XX)
```

```
        RETURNS  THE NAME OFREGION OF THE SCENE IN WHICH THE POINT XX LIES.
ON THE BOUNDARY IS CONSIDERED AS IN AND HENCE ONE OF THE TWO POSSIBLE REGIONS
 WILL BE PICKED ARBITRARILY.


(MSRGTU SCENE TU)
        TU IS A DIRECTED SEGMENT OF THE BOUNDARY GIVEN BY (A B)
        RETURNS THE REGION OF SCENE WHICH HAS TU AS PART OF THE BOUNDARY
        NIL   IF THERE IS NO SUCH REGION


(MSRGKV* SCENE KV*)
        RETURNS THE REGION WHICH HAS KV* AS THE KVERTICES*
        NIL  IF THERE IS NO SUCH REGION


(MSMATCHT SCENE  VS BG1 BG2)
        VS  LIST OF VERTICES TO BE MATCHED
        BG1  LIST OF REGIONS
        BG2  LIST OF REGIONS
                CLEARS THE MEXTE OF T VERTICES OF VS
PUTS PROPER MATES ON MATCHING T*S THAT CONFORM TO FOLLOWING
        1.  DO NOT HAVE MEMBERS OF BG1 AS THE REGION CONTAINING THE STRIGHT LINE AT THE T.
        2.  THE LINE JOINING THE MATCHING T*S DOES NOT INTERSECT BOUNDARY OF MEMBERS OF BG2.
USUALLY BG1 AND BG2 ARE BACKGROUNDS.
BG2 IS NIL IF CONDITION 2 IS NOT REQUIRED.
MSSETUP OR MSSETUP* WILL FIND MATCHING T*S OF SCENE USING THE ONE OPEN REGION FOR BG1
 AND BG2.
WHEN MORE REGIONS ARE DECLARED AS BACKGROUND MATCHING T*S CAN BE UPDATED USING NEW REGIONS FOR BG1 AND B


(MSREPLACE SCENE NLIST)
        NLIST  ((FOR1 WITH1) (FOR2 WITH2)   (FORJ WITHJ))
        UPDATES THE PROPERTY LISTS OF SCENE (AS PRODUCED BY MSSETUP OR MSSETUP*) BY REPLACING
 ALL OCCURENCES OF FORJ BY WITHJ
THE ELEMENTS REPLACED CAN BE NAMES OF VERTICES OR REGIONS OR SGRAFS.
```

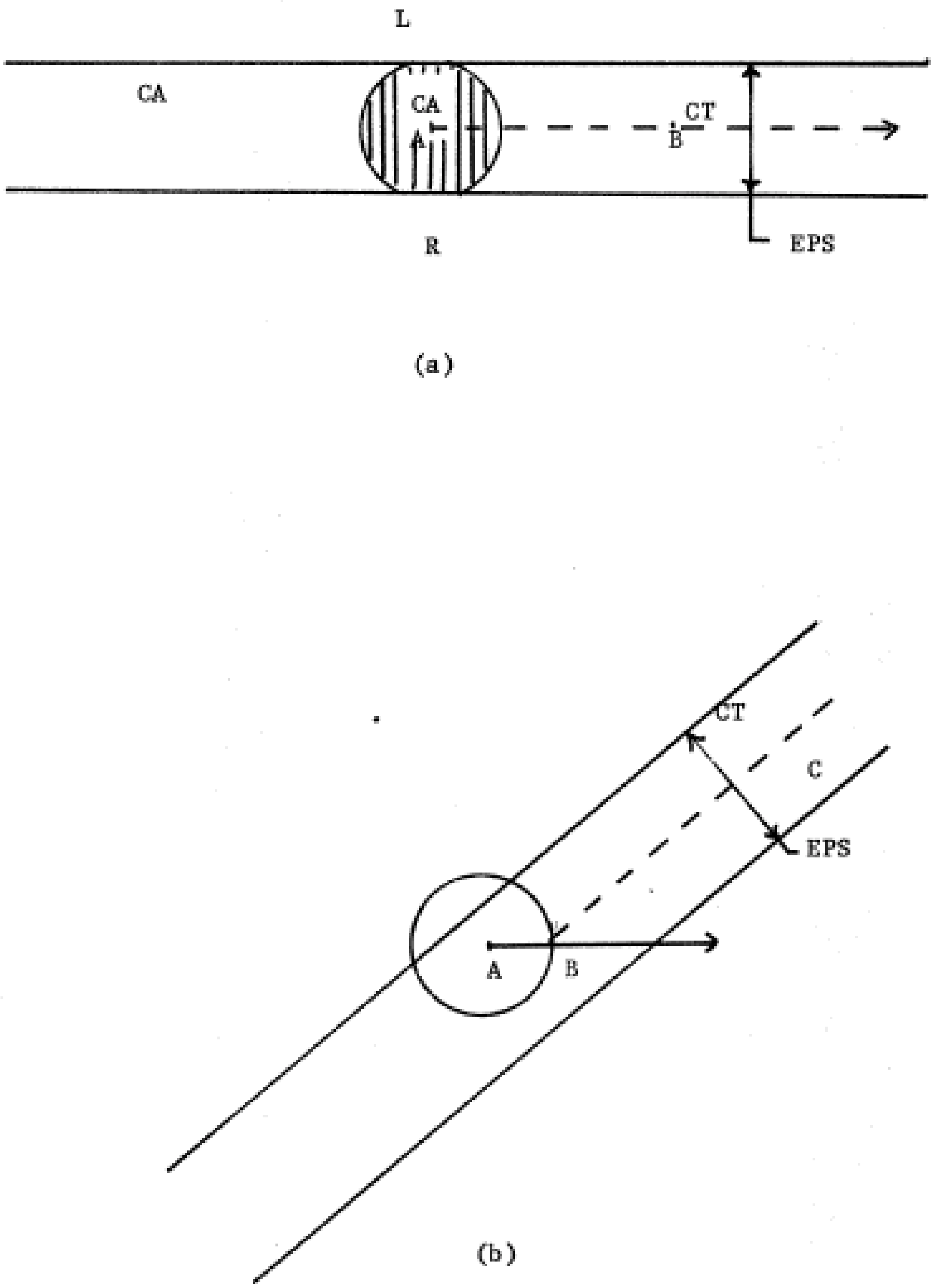


Fig. 1 Orientation of C with respect to AB

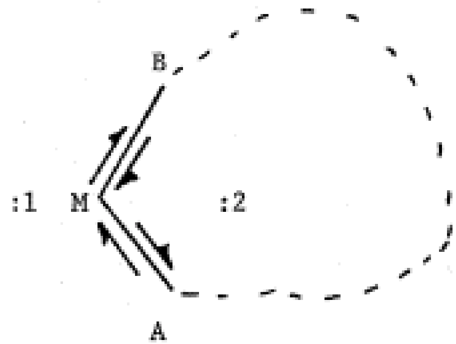


Fig. 2 :1 Unbounded and :2 Bounded.

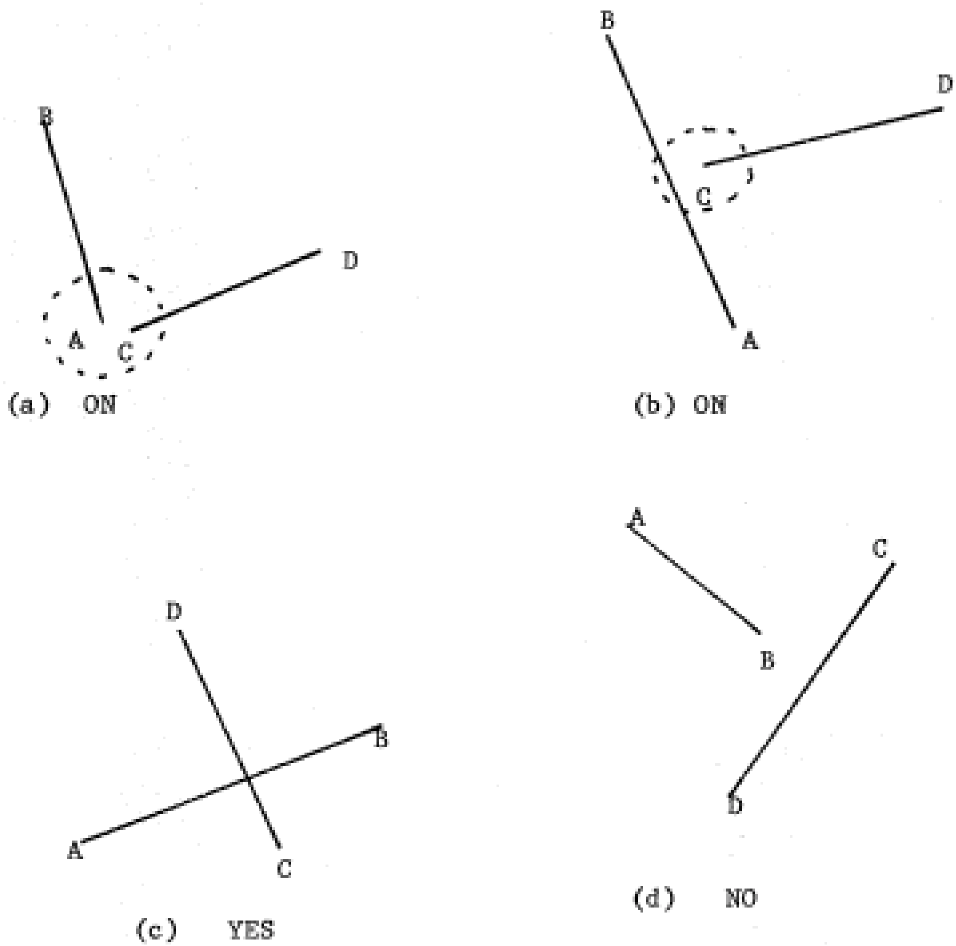


Fig.3 Return of MXSECT

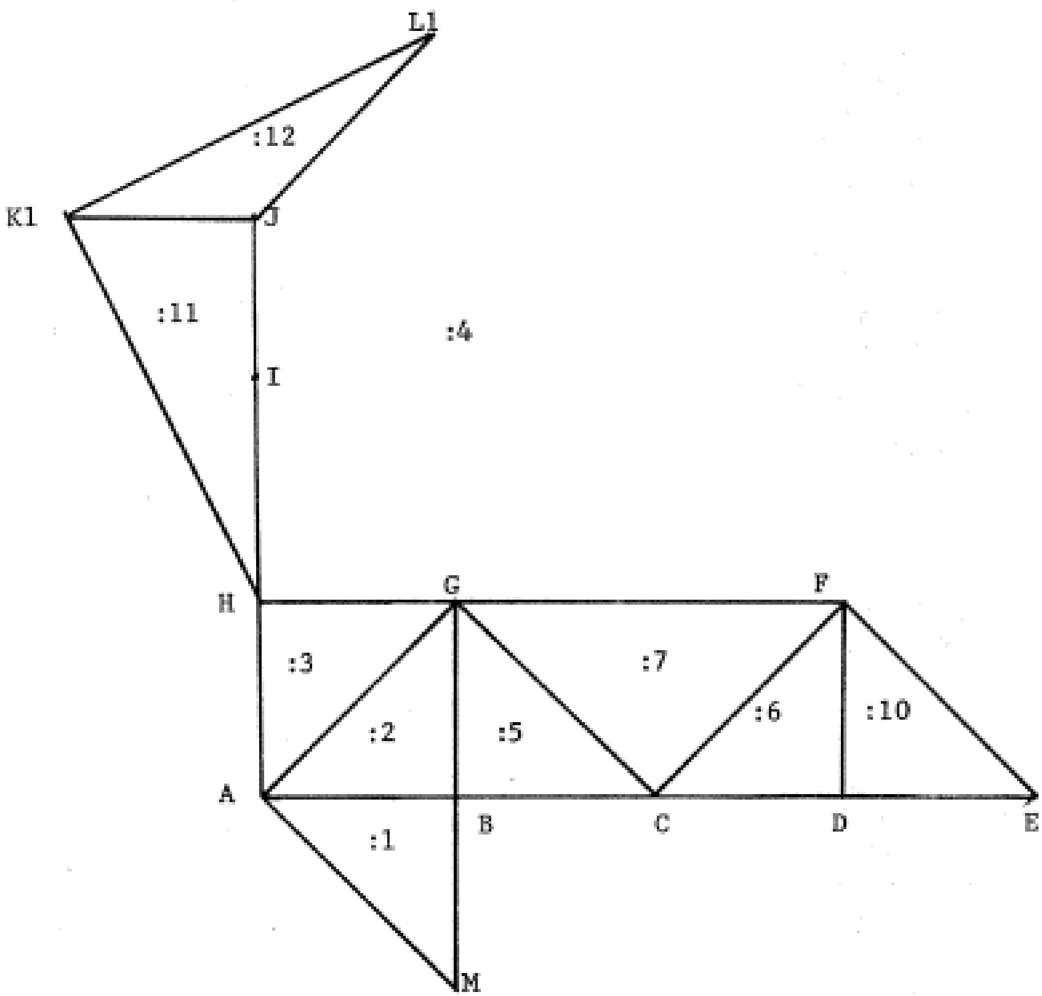


Fig. 'TEST'

(10 (INSIDE 10) (OUTSIDE 10))

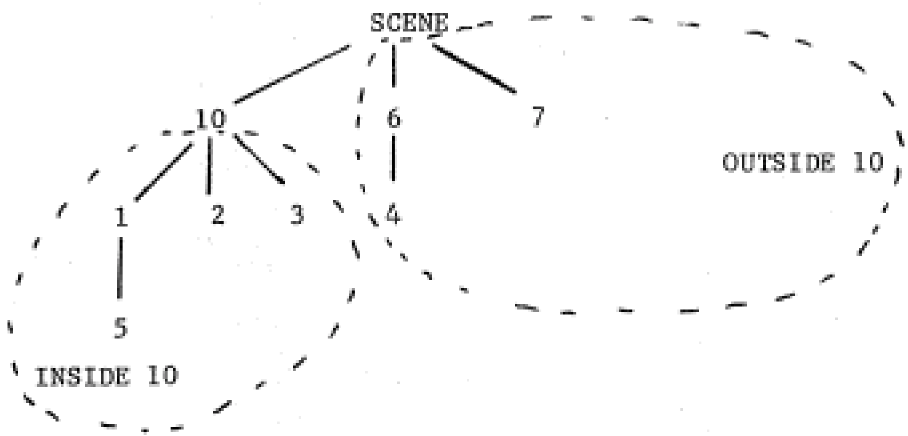
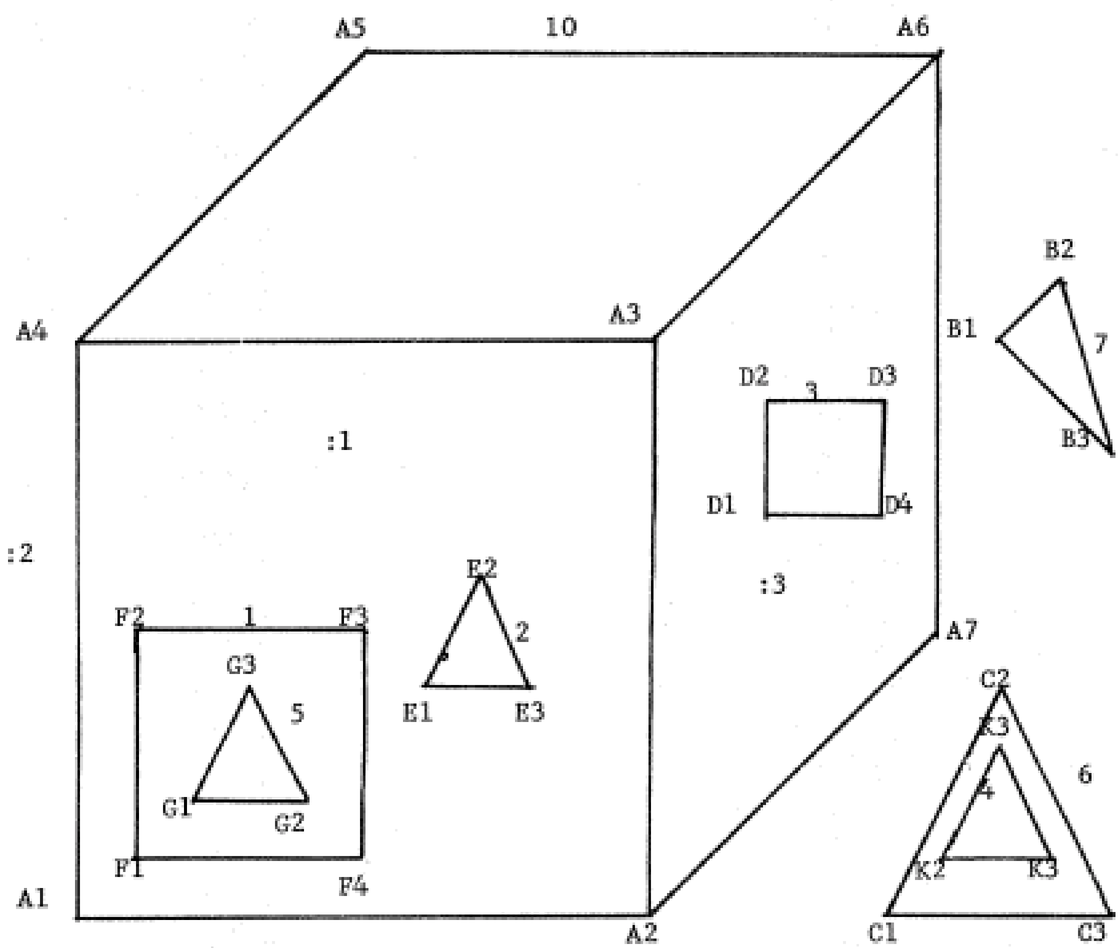
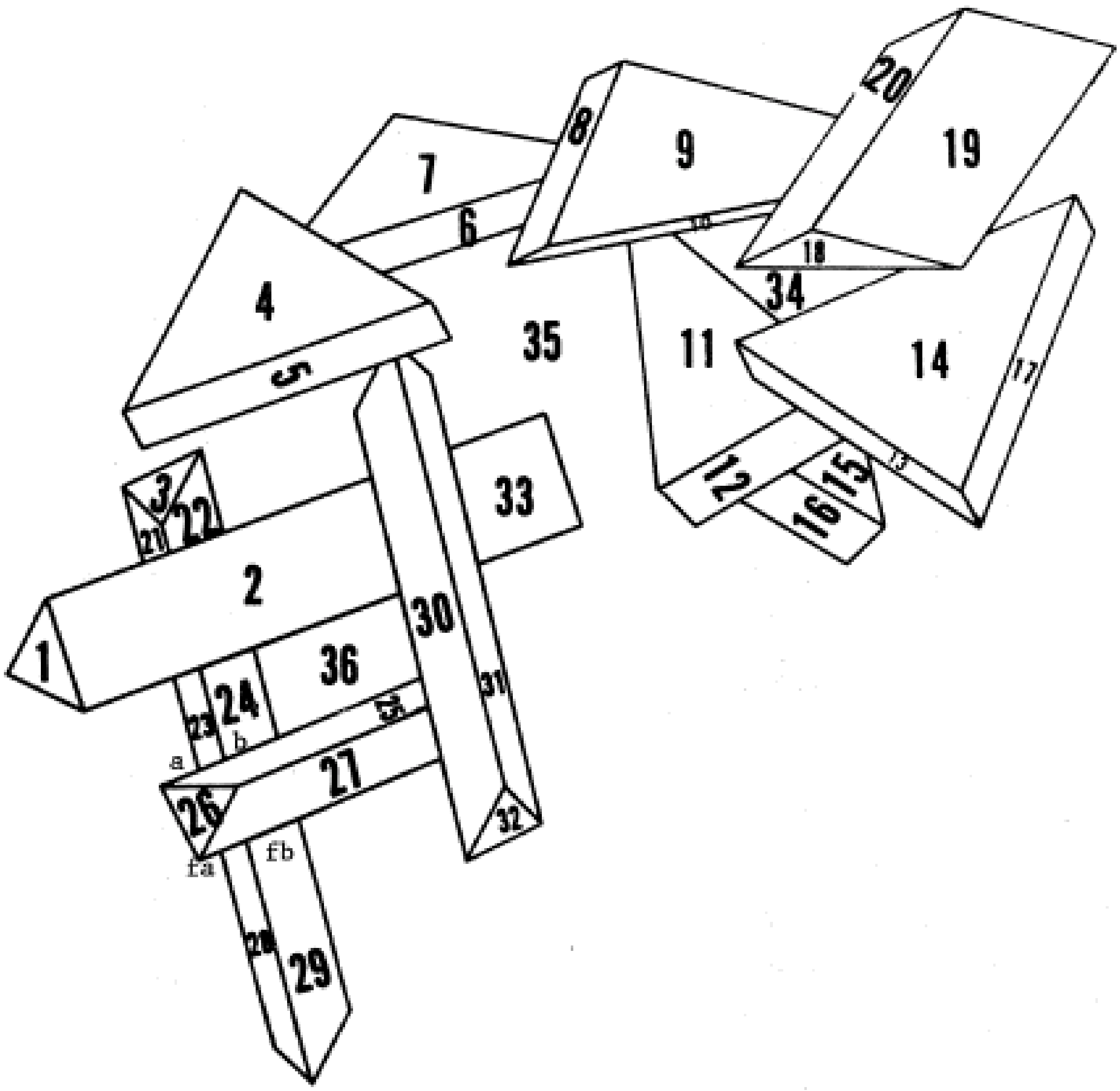


Fig. 'CRAZY'





1
2
3
4
5
6
7
8
9
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
a
b
fa
fb

Fig. 'HARD'

```
SCENE   TEST

PREFIX  NIL

DATA    (TEST (A (15 0) (B H G M)) (B (11 0) (A G C M)) (C (12 0) (B G F D)) (D (13 0) (C F E)) (E (
14 0) (D F)) (F (13 1) (G C D E)) (G (11 1) (H F C B A)) (H (10 1) (A G I K1)) (I (10 2) (H J)) (J (
10 3) (I K1 L1)) (K1 (7 3) (H J L1)) (L1 (11 4) (J K1)) (M (11 -1) (B A)))

TEST

ACCURACY          0.09999999

ERROR   NIL

VERTICES          (A B C D E F G H I J K1 L1 M)

REGIONS           (#12 #11 #10 #7 #6 #5 #4 #3 #2 #1)

CONNECTIVITY      1

SUBGRAPHS         (SGRAF1)

MAXMIN  (K1 E M L1)

BACKGROUND        (#4)

STRUCTURE         (SGRAF1 NIL NIL)

SGRAF1

        MAXMIN(K1 E M L1)

        VERTICES(M A B G C E F D I H L1 K1 J)

        REGIONS(#12 #11 #10 #7 #6 #5 #4 #1 #2 #3)

        BACKGROUND(#4)



CODE FOR VERTICES

VERTEX

        XCOR

        YCOR

        TYPE

        KIND

        NVERTICES

        NREGIONS
```

```
        NEXTE

        COOR
        TYPE=
        OVERTICES

A
        10
        0
        (PEAK (- #4 M))
        (#4 M #1 B #2 G #3 H)
        (M B G H)
        (#4 #1 #2 #3)
        NIL

        (10 0)
        PEAK
        ((M L) (B L) (G L) (H R))
B
        11
        0
        (X (G #5 #2 M #1 #4))
        (#5 G #2 A #1 M #4 C)
        (G A M C)
        (#5 #2 #1 #4)
        NIL

        (11 0)
        CROSS
        ((G L) (A L) (M L) (C L))
C
        12
        0
```

```
(K (#7 #4 D #6 F B #5 G))
(#7 G #5 B #4 D #6 F)
(G B D F)
(#7 #5 #4 #6)
NIL

(12 0)
K
((G L) (B C4) (D L) (F L))
D
13
0
(T (F D E C #10 #6 #4))
(#10 F #6 C #4 E)
(F C E)
(#10 #6 #4)
NIL

(13 0)
T
((F L) (C C4) (E L))
E
14
0
(L (#10 #4))
(#10 D #4 F)
(D F)
(#10 #4)
NIL

(14 0)
L
```

```
((D R) (F L))
F
        13
        1
        (PEAK (E #4 G))
        (#10 E #4 G #7 C #6 D)
        (E G C D)
        (#10 #4 #7 #6)
        NIL

        (13 1)
        PEAK
        ((E R) (G L) (C L) (D L))
G
        11
        1
        (MULTI 3)
        (#7 F #4 H #3 A #2 B #5 C)
        (F H A B C)
        (#7 #4 #3 #2 #5)
        NIL

        (11 1)
        KK
        ((F CA) (H LO) (A L) (B L) (C L))
H
        10
        1
        (K (A #4 #3 I #4 #11))
        (#11 K1 #4 A #3 G #4 I)
        (K1 A G I)
        (#11 #4 #3 #4)
        NIL
```

```
        (10 1)
        X
        ((K1 L) (A _0 (B L) (I L))
I
        10
        2
        (SL (*11 *4))
        (*11 H *4 J)
        (H J)
        (*11 *4)
        NIL

        (10 2)
        SL
        ((H DA) (J DA))
J
        10
        3
        (FORK J)
        (*12 K1 *11 I *4 L1)
        (K1 I L1)
        (*12 *11 *4)
        NIL

        (10 3)
        FORK
        ((K1 L) (I _0 (L1 L))
K1
        7
        3
        (ARROW (J K1 H L1 *11 *12 *4))
        (*12 L1 *4 H *11 J)
```

```
        (L1 H J)
        (*12 *4 *11)
        NIL

        (7 3)
        ARROW
        ((L1 R) (H _3) (J L))
L1
        11
        4
        (L (*12 *4))
        (*12 J *4 K1)
        (J K1)
        (*12 *4)
        NIL

        (11 4)
        L
        ((J R) (K1 _0))
H
        11
        -1
        (L (*1 *4))
        (*4 B *1 A)
        (B A)
        (*4 *1)
        NIL

        (11 -1)
        L
        ((B L) (A R))

CODE FOR REGIONS
```

```
REGION
        PROP
        NEIGHBORS
        KVERTICES
        BACKGROUND

        BOUNDARY
        OTRAVERSE
        NEIGHBORS*
        KVERTICES*
        INSUBGRAPH

R12
        ((R4 L1 R4 K1 R11 J))
        (R4 R4 R11)
        (L1 K1 J)
        NIL

        (((J L1) (L1 K1) (K1 J)))
        (((L1 L) (K1 L) (J L)))
        ((R4 R4 R11))
        ((L1 K1 J))
        NIL

R11
        ((R4 I R4 J R12 K1 R4 H))
        (R4 R4 R12 R4)
        (I J K1 H)
        NIL

        (((H I) (I J) (J K1) (K1 H)))
        (((I GA) (J L) (K1 L) (H L)))
        ((R4 R4 R12 R4))
```

```
((I J KL H))
NIL

#10
((#4 E #4 F #6 D))
(#4 #4 #5)
(E F D)
NIL

(((D E) (E F) (F D)))
(((E L) (F _3) (D L)))
((#4 #4 #6))
((E F D))
NIL

#7
((#6 F #4 G #5 C))
(#6 #4 #5)
(F G C)
NIL

(((C F) (F G) (G C)))
(((F L) (G _3) (C L)))
((#6 #4 #5))
((F G C))
NIL

#6
((#4 D #10 F #7 C))
(#4 #10 #7)
(D F C)
NIL

(((C D) (D F) (F C)))
(((D L) (F _3) (C L)))
```

```
        ((#4 #10 #7))
        ((D F C))
        NIL

#5
        ((#4 C #7 G #2 B))
        (#4 #7 #2)
        (C G B)
        NIL

        (((B C) (C G) (G B)))
        (((C L) (G L) (B L)))
        ((#4 #7 #2))
        ((C G B))
        NIL

#4
        ((#3 H #11 K1 #12 L1 #12 J #11 I #11 H #3 G #7 F #10 E #10 D #6 C #5 B #1 M #1 A))
        (#3 #11 #12 #12 #11 #11 #3 #7 #10 #10 #6 #5 #1 #1)
        (H K1 L1 J I H G F E D C B M A)
        BACKGROUND

        (((A H) (H K1) (K1 L1) (L1 J) (J I) (I H) (H G) (G F) (F E) (E D) (D C) (C B) (B M) (M A)))
)       (((H L) (K1 R) (L1 R) (J L) (I CA) (H L) (G CA) (F R) (E R) (D CA) (C CA) (B L) (M R) (A R))

        ((#3 #11 #12 #12 #11 #11 #3 #7 #10 #10 #6 #5 #1 #1))
        ((H K1 L1 J I H G F E D C B M A))
        (SGRAF1)

#3
        ((#2 G #4 H #4 A))
        (#2 #4 #4)
        (G H A)
        NIL
```

```
(((A G) (G H) (H A)))
(((G L) (H L) (A L)))
((#2 #4 #4))
((G H A))
NIL

#2
((#1 B #5 G #3 A))
(#1 #5 #3)
(B G A)
NIL

(((A B) (B G) (G A)))
(((B L) (G L) (A L)))
((#1 #5 #3))
((B G A))
NIL

#1
((#4 M #4 B #2 A))
(#4 #4 #2)
(M B A)
NIL

(((A M) (M B) (B A)))
(((M L) (B L) (A L)))
((#4 #4 #2))
((M B A))
NIL
```

```
SCENE    CRAZY

PREFIX  NIL

DATA     (CRAZY (A1 (0 0) (A2 A4)) (A2 (12 0) (A1 A3 A7)) (A3 (12 12) (A4 A6 A2)) (A4 (0 12) (A1 A5 A
3)) (A5 (5 17) (A4 A6)) (A6 (17 17) (A5 A3 A7)) (A7 (17 5) (A6 A2)) (B1 (20 12) (B2 B3)) (B2 (21 13)
 (B1 B3)) (B3 (22 15) (B1 B2)) (C1 (16 3) (C2 C3)) (C2 (20 4) (C3 C1)) (C3 (22 0) (C1 C2)) (G1 (2 2)
 (G2 G3)) (G2 (4 2) (G1 G3)) (G3 (3 4) (G1 G2)) (K1 (17 1) (K2 K3)) (K2 (21 1) (K1 K3)) (K3 (20 3) (
K1 K2)) (D1 (14 7) (D2 D4)) (D2 (14 11) (D1 D3)) (D3 (18 11) (D2 D4)) (D4 (18 7) (D3 D1)) (E1 (6 4)
(E2 E3)) (E2 (7 6) (E1 E3)) (E3 (10 4) (E1 E2)) (F1 (1 1) (F2 F4)) (F2 (1 5) (F1 F3)) (F3 (5 5) (F2
F4)) (F4 (5 1) (F3 F1)))

CRAZY

ACCURACY         0.09999999

ERROR   NIL

VERTICES         (A1 A2 A3 A4 A5 A6 A7 B1 B2 B3 C1 C2 C3 G1 G2 G3 K1 K2 K3 D1 D2 D3 D4 E1 E2 E3 F1 F2
 F3 F4)

REGIONS          (#20 #16 #13 #12 #11 #10 #6 #4 #3 #2 #1)

CONNECTIVITY     10

SUBGRAPHS        (SGRAF10 SGRAF7 SGRAF6 SGRAF5 SGRAF4 SGRAF3 SGRAF2 SGRAF1)

MAXMIN  (A4 C3 C3 A6)

BACKGROUND       (#2)

STRUCTURE        (SGRAF10 (SGRAF1 (SGRAF5 NIL NIL) (SGRAF2 NIL (SGRAF3 NIL NIL))) (SGRAF6 (SGRAF4 NIL
 NIL) (SGRAF7 NIL NIL)))

SGRAF10

         MAXMIN(A4 A6 A2 A5)

         VERTICES(A1 A7 A2 A6 A5 A4 A3)

         REGIONS(#4 #3 #2 #1)

         BACKGROUND(#2)

SGRAF7

         MAXMIN(B1 B3 B3 B2)

         VERTICES(B3 B2 B1)

         REGIONS(#6 #2)

         BACKGROUND(#2)

SGRAF6

         MAXMIN(C1 C3 C1 C2)
```

```
        VERTICES(C3 C2 C1)
        REGIONS(*10 *2)
        BACKGROUND(*2)
SGRAF5
        MAXMIN(G1 G2 G1 G3)
        VERTICES(G3 G2 G1)
        REGIONS(*12 *11)
        BACKGROUND(*12)
SGRAF4
        MAXMIN(K1 K2 K1 K3)
        VERTICES(K3 K2 K1)
        REGIONS(*10 *13)
        BACKGROUND(*10)
SGRAF3
        MAXMIN(D1 D3 D1 D2)
        VERTICES(D4 D3 D2 D1)
        REGIONS(*16 *3)
        BACKGROUND(*3)
SGRAF2
        MAXMIN(E1 E3 E1 E2)
        VERTICES(E3 E2 E1)
        REGIONS(*20 *1)
        BACKGROUND(*1)
SGRAF1
        MAXMIN(F1 F3 F1 F2)
        VERTICES(F4 F3 F2 F1)
        REGIONS(*12 *1)
        BACKGROUND(*1)
```

```
*2
        ((*1 A4 *4 A5 *4 A6 *3 A7 *3 A2 *1 A1) (*6 B2 *6 B3 *6 B1) (*10 C2 *10 C3 *10 C1))
        (*1 *4 *4 *3 *3 *1 *6 *6 *6 *10 *10 *10)
        (A4 A5 A6 A7 A2 A1 B2 B3 B1 C2 C3 C1)
        BACKGROUND

        (((A1 A4) (A4 A5) (A5 A6) (A6 A7) (A7 A2) (A2 A1)) ((B1 B2) (B2 B3) (B3 B1)) ((C1 C2) (C2 C3
) (C3 C1)))
        (((A4 R) (A5 R) (A6 R) (A7 R) (A2 R) (A1 R)) ((B2 R) (B3 R) (B1 R)) ((C2 R) (C3 R) (C1 R)))
        ((*1 *4 *4 *3 *3 *1) (*6 *6 *6) (*10 *10 *10))
        ((A4 A5 A6 A7 A2 A1) (B2 B3 B1) (C2 C3 C1))
        (SGRAFF7 SGRAF6 SGRAF10)

*1
        ((*2 A2 *3 A3 *4 A4 *2 A1) (*20 E2 *20 E3 *20 E1) (*12 F2 *12 F3 *12 F4 *12 F1))
        (*2 *3 *4 *2 *20 *20 *20 *12 *12 *12 *12)
        (A2 A3 A4 A1 E2 E3 E1 F2 F3 F4 F1)
        NIL

        (((A1 A2) (A2 A3) (A3 A4) (A4 A1)) ((E1 E2) (E2 E3) (E3 E1)) ((F1 F2) (F2 F3) (F3 F4) (F4 F1
)))
        (((A2 L) (A3 L) (A4 L) (A1 L)) ((E2 R) (E3 R) (E1 R)) ((F2 R) (F3 R) (F4 R) (F1 R)))
        ((*2 *3 *4 *2) (*20 *20 *20) (*12 *12 *12 *12))
        ((A2 A3 A4 A1) (E2 E3 E1) (F2 F3 F4 F1))
        (SGRAF2 SGRAF1)
```