*Reprinted from*
*AFIPS – Vol.33*

*Thompson Book Co.*
*National Press Building*
*Washington, D. C.*

# Decomposition of a visual scene into three-dimensional bodies

*by* ADOLFO GUZMAN

*Massachusetts Institute of Technology*
Cambridge, Massachusetts

## INTRODUCTION

We consider visual scenes composed by the optical image of a group of bodies. When such a scene is "seen" by a computer through a film spot scanner, image dissector, or similar device, it can be treated as a two-dimensional array of numbers, or as a function of two variables.

At a higher level, a scene could be meaningfully described as a conglomerate of points, lines and surfaces, with properties (coordinates, slopes, ...) attached to them.

Still a more sophisticated description could use terms concerning the bodies or objects which compose such a scene, indicating their positions, inter-relations, etc.

This paper describes a program which finds bodies in a scene, presumably formed by three-dimensional objects. Some of them may not be completely visible. The picture is presented as a line drawing.

When SEE—the pretentious name of the program—analyzes the scene **TRIAL** (see Figure 1 'TRIAL'), the results are:

(BODY 1. IS :6 :2 :1)

(BODY 2. IS :11 :12 :10)

(BODY 3. IS :4 :9 :5 :7 :3 :8 :13)

SEE looks for three-dimensional objects in a two-dimensional scene. The scene itself is not obtained from a visual input device, or from an array of intensities or brightness. Rather, it is assumed that a preprocessing of some sort has taken place, and the scene to be analyzed is available in a symbolic format (to be described in a later Section), in terms of points (vertices), lines (edges), and surface (regions).

SEE does not have a pre-concieved idea of the form or model of the objects which could appear in a given scene. The only supposition is that the bodies are solid objects formed by plane surfaces; in this way, it can not find "cubes" or "houses" in a scene, since it does not know what a "house" is. Once SEE has partitioned a scene into bodies, some other program will work on them and decide which of those bodies are "houses."

Thus, SEE is intended to serve as a link between a pre-processor [1,2] which transforms intensity pictures into point or line pictures, [5] and a recognizer (such as TD [3] or DT [4]), which handles this line picture and finds bodies, objects or zones matching with certain patterns or models. Instead of searching through the whole scene looking for parts to match its models, the work of the recognizer becomes simpler after SEE has partitioned the scene into bodies, because the data to be searched (matched) are smaller and better organized.

The analysis which SEE makes of the different scenes generally agrees with human opinion, although in some ambiguous cases it behaves rather conservatively. Distributed over these pages, the reader will find examples of scenes analyzed by SEE, and the peculiarities and behavior of the program will become clear.

The program SEE, written in LISP, has been tested in the PDP-6 machine of the Artificial Intelligence Group, Project MAC, at Massachusetts Institute of Technology. A preliminary version, written in CONVERT, [6] was used extensively for a quick test of ideas which shaped the program to its actual form. The analysis of a scene takes from 30 to 90 seconds, with the program running interpreted under the interpreter of the LISP programming system.

A more technical description of SEE can be found in an unpublished memorandum. [7]

### Related work

Rudd H. Canaday [8] in 1962 analyzed scenes composed of two-dimensional overlapping objects, "straight-

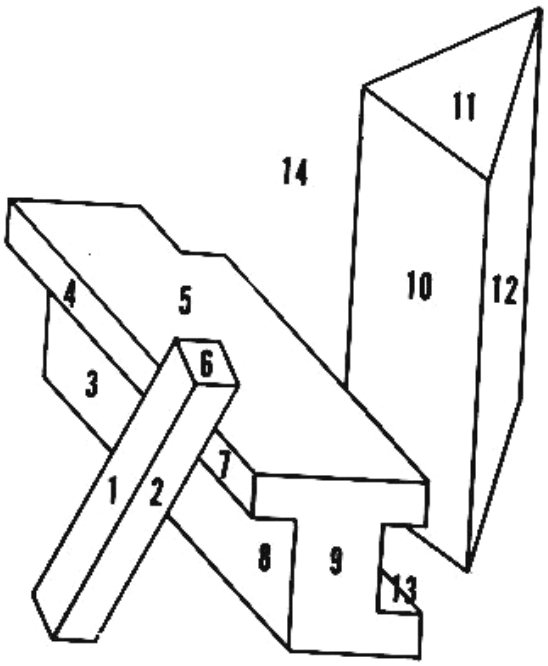



FIGURE 1—TRIAL

The program analyzes this scene and finds 3 bodies:

(BODY 1 IS :6 :2 :1)
(BODY 2 IS :11 :12 :10)
(BODY 3 IS :4 :9 :5 :7 :3 :8 :13)

sided pieces of cardboard." His program breaks the image into its component parts (the pieces of cardboard), describes each one, gives the depth of each part in the image (or scene), and states which parts cover which.

Roberts [9] in 1963 described programs that (1) convert a picture (a scene) into a line drawing and (2) produce a three-dimensional description of the objects shown in the drawing in terms of models and their transformations. The main restriction on the lines is that they should be a perspective projection of the surface boundaries of a set of three-dimensional objects with planar surfaces. He relies on perspective and numerical computations, while SEE uses a heuristic and symbolic (i.e., non-numerical ) approach. Also, SEE does not need models to isolate bodies. Roberts' work is probably the most important and closest to ours.

Actually, several research groups (at Massachusetts Institute of Technology, [10] at Stanford University, [11] at Stanford Research Institute [12]) work actively towards the realization of a mechanical manipulator, i.e., an intelligent automata who could visually perceive and successfully interact with its enviornment, under the control of a computer. Naturally, the mechanization of visual perception forms part of their research, and important work begins to emerge from them in this area.

**Organization of the paper**

It is formed by the following headings:

- Introduction and related previous work.
- Input Format. The representation of the scene as it is entered to the computer.
- Format of a Scene. The representation of the scene as SEE expects.
- Type of Vertices. Classification of vertices according to their topology.
- The program. Analysis of the algorithm, description of heuristics.
- Interesting examples. Discussion. Future work.

*Input format*

For testing purposes, the scenes are entered by hand in a simplified format (called input format), and then some routines convert this to the form required by SEE. Eventually, the data will come from a visual input device, through a preprocessor.[2,5]

**Examples of a scene**

Suppose we want to describe the scene 'CUBE.' We begin by giving (in LISP) a value to 'CUBE.' (See Figure 2 'Cube')

```
(SETQ CUBE (QUOTE (A 1.0 1.0 (:1 B :4 G)
                   B 1.0 5.0 (:1 E :2 C :4 A)
                   C 3.0 7.0 (:2 D :4 B)
                   D 8.0 7.0 (:2 E :3 F :4 C)
                   E 6.0 5.0 (:2 B :1 G :3 D)
                   F 8.0 3.0 (:3 G :4 D)
                   G 6.0 1.0 (:1 A :4 F :3 E)
                   )))
```

Thus we associate with each vertex its coordinates and a list, in counterclockwise order, of regions and vertices radiating from that vertex.

The conversion of the scene, as just given, into the form which SEE expects, is made by the function LLENA; thus, (LLENA CUBE) will put in the prop-

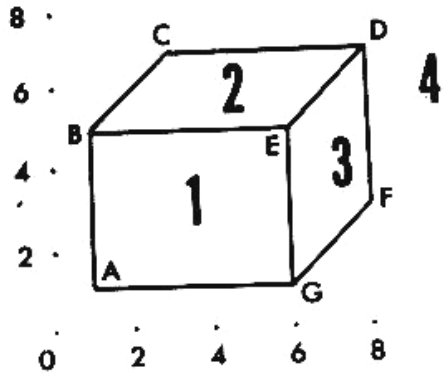



FIGURE 2—'CUBE'—A scene.

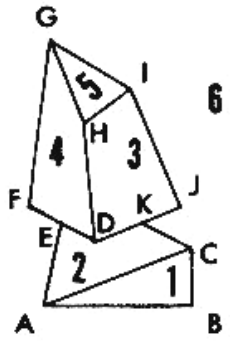



FIGURE 3—'ONE.' A scene. Vertices and surfaces (regions) are the main components of a scene.

erty list of CUBE the properties REGIONS and VERTICES; in the property list of each vertex, the properties XCOR, YCOR, NREGIONS, NVERTICES and KIND are placed; and in the property list of each region, it places the properties NEIGHBORS and KVERTICES. It also marks region :4 as part of the background.

In other words, LLENA converts a scene from the 'Input Fornat' to the 'Format of a Scene' described in the next section.

*Format of a scene*

A scene is presented to our program as a scene in a special symbolic format, which we now describe. Essentially, it is an arrangement of relations between vertices and regions.

A scene has a name which identifies it; this name is an atom whose property list contains the properties 'REGIONS,' 'VERTICES' and 'BACKGROUND.' For example, the scene ONE (see Figure 3 'ONE') has the name 'ONE.' In the poperty list of 'ONE' we find

REGIONS —(:1 :2 :3 :4 :5 :6)
Unordered list of regions composing scene ONE.

VERTICES —(A B C D E F G H I J K)
Unordered list of vertices composing the scene ONE.

BACKGROUND— (:6)
Unordered list of regions composing the background of the scene ONE.

**Region**

A region corresponds to a surface limited by a simple connected curve. For instance, in ONE, the surface delimited by the vertices A B C is a region, called :1, but G I J K D H is not.

Each region has as name an atom which possesses additional properties describing different attributes of the region in question. These are 'NEIGHBORS,' 'KVERTICES,' and 'FOOP'. For example, the region in scene ONE formed by the lines AE, ED, DK, KC, CA, has ':2' at its name. In the property list of :2 we find:

NEIGHBORS — (:3 :4 :6 :1 :6)
Counterclockwise ordered list of all regions which are neighbors to :2. For each region, this list is unique up to cyclic permutation.

KVERTICES — (D E A C K)
Counterclockwise ordered list of all vertices which belong to the region :2. This list is unique up to cyclic permutation.

FOOP — (:3 D :4 E :6 A :1 C :6 K)
Counterclockwise ordered list of alternating neighbors and kvertices of :2. This list is unique up to cyclic permutation.

The FOOP property of a region is formed by a man who walks on its boundary always having this region to his left, and takes note of the regions to his right and of the vertices which he finds in his way.

### Vertex

A vertex is the point where two or more lines of the scene meet; for instance, A, G, and K are vertices of the scene ONE.

Each vertex has as name an atom which posseses additional properties describing different attributes of the vertex in question. These are 'XCOR,' 'YCOR,' 'NVERTICES,' 'NREGIONS,' and 'KIND.' For example, the vertex H (see scene ONE) has in its property list:

XCOR — 3.0
x-coordinate

YCOR — 15.0
y-coordinate

NVERTICES — (I G D)
Counterclockwise ordered list of vertices to which H is connected. This list is unique up to cyclic permutation.

NREGIONS — (:3 :5 :4)
Counterclockwise ordered list of regions to which H belongs. This list is unique up to cyclic permutation.

KIND — (:3 I :5G :4 D)
Counterclockwise ordered list of alternating nregions and nvertices of H. This list is unique up to cyclic permutation.

The KIND property of a vertex is formed by a man who stands at the vertex and, while rotating counterclockwise, takes note of the regions and vertices which he sees.

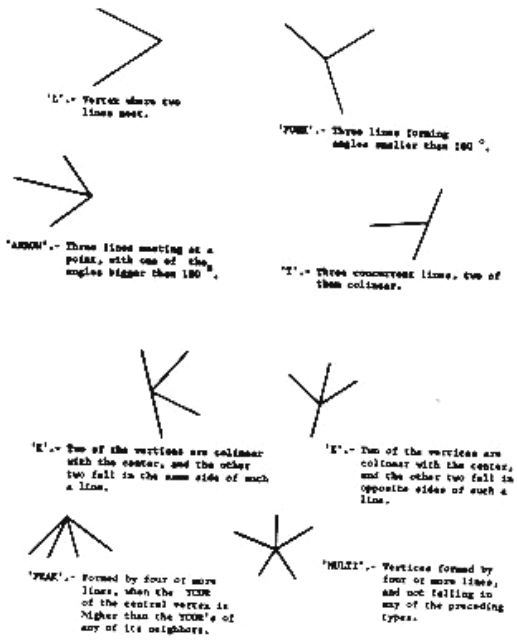
NREGIONS and NVERTICES are then easily derived from KIND: take the odd positioned elements of KIND, and its even-positioned elements, respectively.

*Types of vertices*

Vertices are classified according to the slope, disposition and number of lines which form them. The function (TYPEGENERATOR L), where L is a list of vertices, performs this classification, putting in the property list of each one of the elements of L, the type to which it belongs.

The TYPE of a vertex is always a list of two elements; the first is the *type-name*: one of 'L,' 'FORK,' 'ARROW,' 'T,' 'K,' 'X,' 'PEAK,' 'MULTI'; the second element is the *datum*, which generally is a list, whose form varies with the *type-name* and contains information in a determined order about the vertex in question. (See Table I 'VERTICES').

TABLE I—'VERTICES' Classification of vertices of a scene.




*The program*

The program named SEE accepts a scene expressed in the notation described above and produces outputs lists identifying and describing the bodies present in the scene.

In this section we describe the program, and how it achieves its goals, by discussing the procedures, heuristics etc., employed and the way they work. We begin with several examples.

Example 1. Scene **'STACK'** This scene (see Figure 4 'STACK') is analyzed by SEE, with the following results:

```
(LOCAL ASSUMES (:5) (:13 :14) SAME BODY)
    (BODY 1. IS :1 :3 :2 :18 :17)
    (BODY 2. IS :4 :16 :15)
    (BODY 3. IS :7 :6 :11 :12)
    (BODY 4. IS :9 :8 :10)
    (BODY 5. IS :13 :14 :5)
    (BODY 6. IS :20 :19)
```

Results for scene STACK

Example 2. Scene **'BRIDGE.'** With this example,

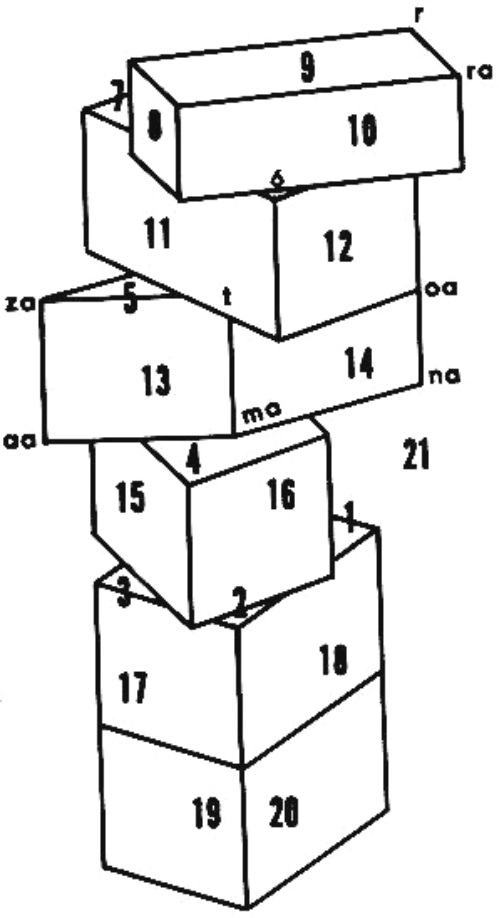



FIGURE 4—'STACK.' This scene is analyzed by SEE, with results detailed in Example 1. All bodies are correctly found. Some of the vertices appear in the drawing with their names; in other drawings we will omit the names; we are also omiting the coordinate axes.

we will give details of the program's operation. We start by loading into LISP the programs for SEE and its auxiliary functions.
Then, we evaluate:

(UREAD BRIDGE SCEN3) ↑Q
This causes scene BRIDGE (see Figure 5 'BRIDGE'), to be read (in the format described previously) and transformed to the proper form which SEE expects (also described before).

Finally, we evaluate

(SEE (QUOTE BRIDGE))
This calls SEE to work on BRIDGE.

Results appear in Table II, 'RESULTS FOR BRIDGE.'

(LOCAL ASSUMES (:18) (:19) SAME BODY)
(LOCAL ASSUMES (:28) (:29) SAME BODY)
(LOCAL ASSUMES (:10) (:8 :11 :5 :6 :4) SAME BODY)
(LOCAL ASSUMES (:7) (:8 :11 :5 :6 :4 :10) SAME BODY)

(SINGLEBODY ASSUMES (:19 :18) (:16) SAME BODY)

RESULTS
(BODY 1. IS :24 :9 :21 :27 :12 :25)
(BODY 2. IS :22 :26 :23)
(BODY 3. IS :17 :3 :20)
(BODY 4. IS :1 :2) Results for scene 'BRIDGE'
(BODY 5. IS :14 :15 :13)
(BODY 6. IS :19 :18 :16)
(BODY 7. IS :29 :28)
(BODY 8. IS :8 :11 :5 :6 :4 :10 :7)

TABLE II—RESULTS FOR BRIDGE

SEE finds eight bodies in scene 'BRIDGE' (See Figure 5 'BRIDGE'). Unnecessary detail has been removed from the drawings; the reader must bear in mind that the complete scene contains vertices (whose names do not appear in the drawings) and their coordinates (which are also not shown), as well as edges and regions.

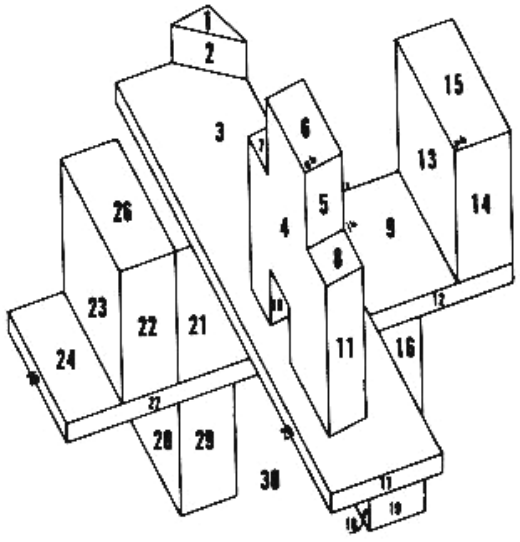



FIGURE 5—'BRIDGE.' The long body :25 :24 :27 :21 :9 :12 is correctly identified. (See Table II)

### SEE and its parts

The operation of SEE is quite straightforward; the program is not recursive and does not do any tree search. Its main parts, which are executed one after another, unless otherwise indicated, are:

FILLSCENE. The properties SLOP and TYPE are generated for each vertex, if they were not already present.

CLEAN. Old properties used internally by SEE are removed from vertices and regions.

EVIDENCE. An analysis is made of vertices, regions and associated information, in search of clues that indicate that two regions form part of the same body. If evidence exists that two regions in fact belong to the same body, they are linked or marked with a "gensym" (both receive the same new label). There are two kinds of links, called strong (global) or weak (local).

LOCALEVIDENCE. Some features of the scene will weakly suggest that a group of regions should be considered together, as part of the same body. This part of the program is that which produces the 'local' links or evidence.

GLOBAL. The 'strong' links gathered so far are analyzed; regions are grouped into "nuclei" of bodies, which grow until some conditions fail to be satisfied (a detailed explanation follows later).

LOCAL. Weak evidence is taken into account for deciding which of the unsatisfactory global links should be considered satisfactory, and the corresponding nuclei of bodies are then joined to form a single and bigger nucleus. Assumptions made by LOCAL are printed (see output of SEE). LOCAL may call GLOBAL again, or go on.

SINGLEBODY. If a single region does not belong to a larger nucleus, but is linked by one strong evidence to another region, it is incorporated into the nucleus of that other region. This part of the program may call GLOBAL or LOCAL again, if necessary, or continue. SINGLEBODY also prints its assumptions.

RESULTS. The regions belonging to the background are screened out, and the results are printed.

### Operation

Here is explained in considerable detail each of the parts of SEE that have been sketched above. This will help the reader understand the behavior of the program, its strength and deficiencies.

Example. Scene **'TOWER.'** First, we begin by showing a typical analysis of SEE with a somewhat complicated scene (see Figure 6 'TOWER'). Most of the scenes contain several "nasty" coincidences: a vertex of an object lies precisely on the edge of another object; two nearly parallel lines are merged into a single one, etc. This has been done on purpose, since a non-sophisticated preprocessor will tend to make this kind of error.

The output is given in Table III, 'RESULTS FOR TOWER.'

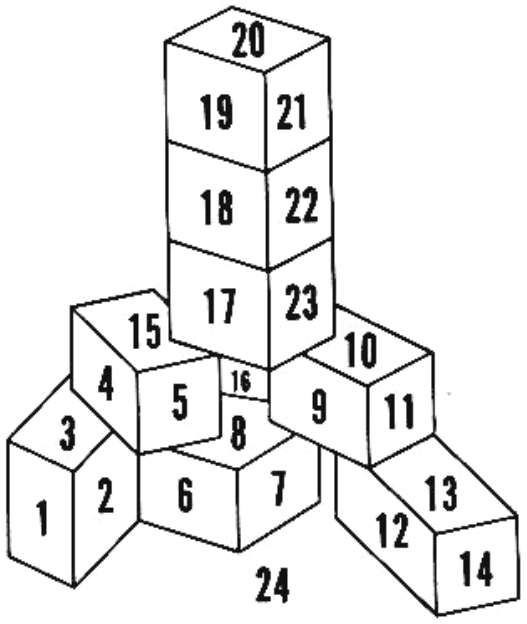



FIGURE 6—'TOWER.' Neither LOCAL nor SINGLEBODY are necessary to correctly parse this scene into the bodies which form it. There are many 'strong' links in this scene and SEE makes good use of them. (See Table III)

| | |
|---|---|
| (BODY 1. IS :3 :2 :1)<br>(BODY 2. IS :5 :15 :4)<br>(BODY 3. IS :23 :17)<br>(BODY 4. IS :6 :7 :8)<br>(BODY 5. IS :10 :11 :9)<br>(BODY 6. IS :13 :14 :12)<br>(BODY 7. IS :18 :22)<br>(BODY 8. IS :20 :19 :21) | Results for TOWER |

TABLE III—Results for Figure 6 'TOWER'

FILLSCENE, CLEAN. These two parts of SEE are simple; if necessary, FILLSCENE calls SLOPGENERATOR and TYPEGENERATOR; CLEAN removes some unwanted properties.

EVIDENCE. Strong or global links are placed by this part of the program. Two functions are used: EVERTICES and TJOINTS.

EVERTICES. This function considers each vertex of the scene under the following rules:

L. Vertices of the type 'L' are not considered.

FORK. If three regions meet at a vertex of the FORK type (Table IV (b)), and none of them is the background, links between them will be formed. For instance in Figure 6 'TOWER,' we establish the links :19— :20, :19— :21, and :20— :21. Nevertheless, some of these links may not be produced: in Figure 7 'FORK,' the link between :3 and :2 is *not* produced because Q is a "passing T"; the link between :1 and :3 is *not* generated because R is an 'L.' The link between :1 and :2 *is* generated.

This heuristic is powerful, and (see Figure 5 'BRIDGE') allows us, for instance, while working on vertex jb, to omit the link between regions :5 and :9 in scene 'BRIDGE.' Nevertheless, this same heuristic puts a link between regions :8 and :9 of the same scene. As we shall see later, this situation is not too bad.

ARROW. This type of vertex (Table IV (c)) causes two of its regions (the 'left' and the 'right' one) to be joined; in Table IV (c), we will put a link between :1 and :2, which counts as evidence that :1 and :2 belong to the same body. Nothing is said about :3.

For instance, this type of vertex joints 1: and :2 in Figure 5 'BRIDGE.'

X. Two cases are distinguished:

(a) The X is formed by the intersection of two lines. No evidence is formed.

(b) Otherwise (Table IV (f)), links :1— :2 and :3— :4 are formed. This type of X occurs when we have piles of objects; for instance, in Figure 6 'TOWER,' :18 and :22 are considered as belonging to the same body, due to this type of vertex.

PEAK. All its regions (table IV (h)), except the one containing the obtuse angle, are linked to each other. See also Figure 14 'CORN.'

T. A search is made for another T to match the vertex currently analyzed; two T's match if they are colinear and "facing each other;" if there are several pairs, the closest is chosen. For instance (Table IV (d)), A and B are paired. An indicator 'NEXTE' is placed in such vertices.

(a) Once the NEXTE is determined, EVERTICES establishes a link between :1 and :2 (Table IV (d)), and another between :3 and :4. These links will not be produced if the results of them is to associate the background with something that is not the background.

(b) The following test is done (Figure 8 'PARALLEL'): If neither :1 or :2 is the background, but both have it as neighbor, and in some part of the boundary of :1 (and the same holds for :2) with the background, the segment them (M — K in Figure 8 'PARALLEL') is parallel to the central segment (O—P) of the T, then :1 and :2 are linked. For instance, in Figure 4 'STACK,' this analysis applied to the vertex T will produce evidence that :13 and :14 belong to the same body (since ZA—AA, T—MA and OA—NA are parallel). This is a rather global heuristic although only useful for bodies with parallel faces. Also, EVERTICES classifies T's according to the slope of their central segments.

A summary of the strong links put by EVERTICES is found in Table IV 'GLOBAL EVIDENCE.'

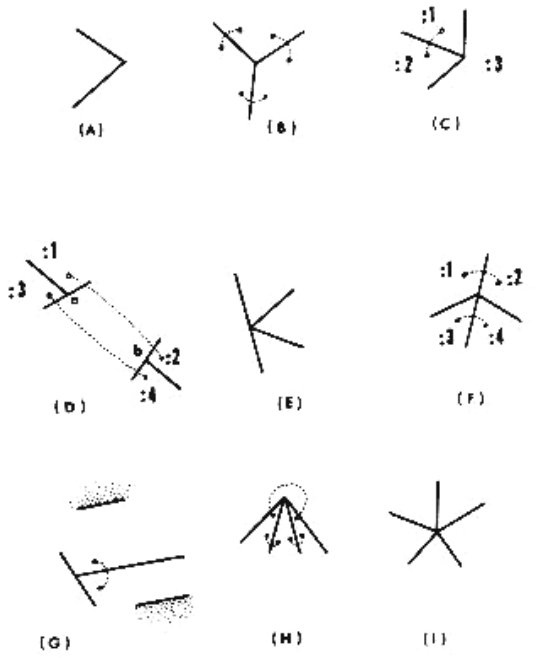
TJOINTS. This function actuates on T's and established global evidence as described in part (a) of T (Table IV (d)).

LOCALEVIDENCE. An arrow is a leg if one of its left or right vertices is a corner (if necessary, through a chain of matched T's) which has a side parallel to the central segment of the arrow. The function LEGS finds legs in the scene, and stores this information in a list of 'weak' links (see Figure 9 'LEGS').

GLOBAL. Strong evidence is analyzed in this part of the program. When this section is entered, several links (strong and weak) exist among the different regions of the scene. These links are shown pictorially in Eigure 10 'LINKS,' for the scene 'BRIDGE' (see both). All the links to the background (:30) are deleted: the background cannot be part of any body.

*Definition:* a *nucleus* (of a body) is either a region or a set of nuclei which has been formed by the following rule.

TABLE IV—'GLOBAL EVIDENCE' The strong links are represented by dotted lines



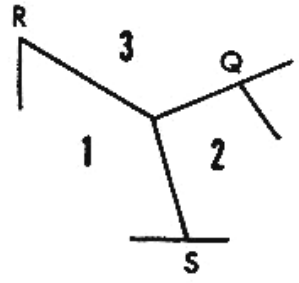




FIGURE 7—'FORK.'

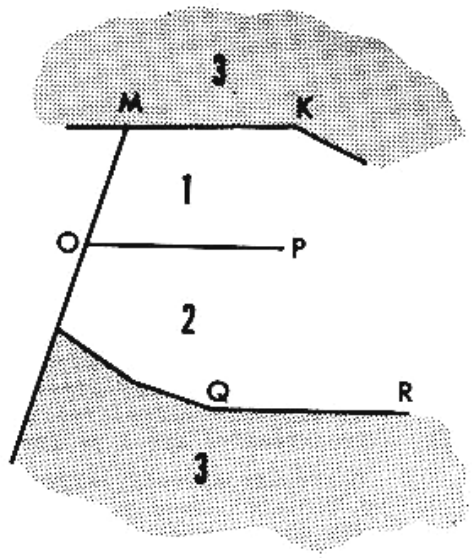



FIGURE 8—'PARALLEL.' A link is established between :1 and :2 because they do not belong to the background, but the background is a neighbor of both of them, and the segment that separates :1 from the background (and the same is true for :2) is parallel to OP, the central segment of the T in question (:3 is the background).

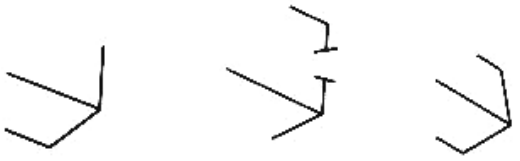


FIGURE 9—'LEGS.'
Three different types of legs.

*Rule:* If two nuclei are connected by two or more links, they are merged into a larger nucleus by concatenation.

(Two nuclei A and B are linked if regions a and b are linked where a $\epsilon$ A and b $\epsilon$ B). For instance, regions :8 and :11 are put together, because there exists two links among them, to form the nucleus :8—11. Now, we see that region :4 (see Figure 10 'LINKS') has two links with this nucleus :8—11, and therefore the new nucleus :8—11 = 4 is formed.

We let the nuclei grow and merge under the former rule, until no new nuclei can be formed. When this is the case, the scene has been partitioned into several "maximal" nuclei; between any two of these there are zero or, at most, one link. For example, figure 10 'LINKS' will be transformed into Figure 11 'NUCLEI.'

LOCAL. If some strong link joining two "maximal" nuclei is also reinforced by a weak link, these nuclei are merged.

For example, in scene BRIDGE (see Figure 5 'BRIDGE') the following local links exist (among others): :9 to :4, :10 to :4, :28 to :29, :18 to :19. Therefore, the corresponding nyclei are merged and now figure NUCLEI is transformed into figure 'NEW NUCLEI.'

A weak link does not cause the regions which it links to be merged or considered as belonging to the same body unless there is, in addition, one strong evidence between such regions. LOCAL may call GLOBAL again, if necessary.

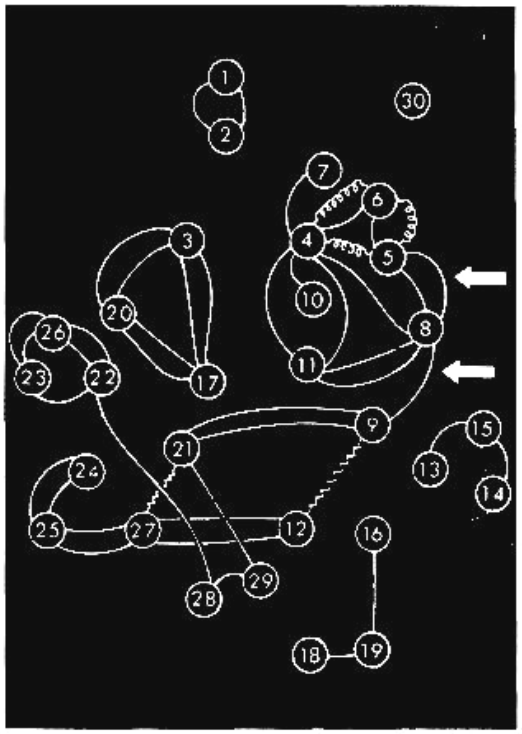



FIGURE 10—'LINKS.' This figure represents scene 'BRIDGE,' with the strong links between its regions (represented here by circles) shown. The dotted links represent the evidence generated by the vertex PB (see Figure 5 'BRIDGE'). The short arrows show the links put by vertex JB; note that a link between :5 and :9 is not put, because S (see Figure 5 'BRIDGE') is a passing t-joint. Zig-zag links are produced by the mechanism described in part (b) of T. (See Figure 8 'PARALLEL'). Curled links are produced by vertex GB; even if this vertex were occluded, and the links were missing, there is still enough evidence to identify regions :4 :5 and :6 as belonging to the same body. Weak links are not shown.

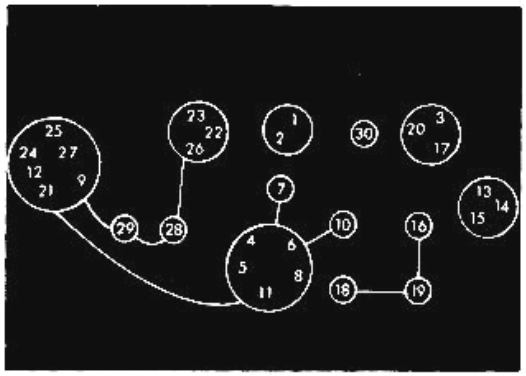



FIGURE 11—'NUCLEI.' After joining all nuclei having two or more links in common, the representation for the scene 'BRIDGE' changes from that shown in figure 10 'LINKS' to the one shown here.

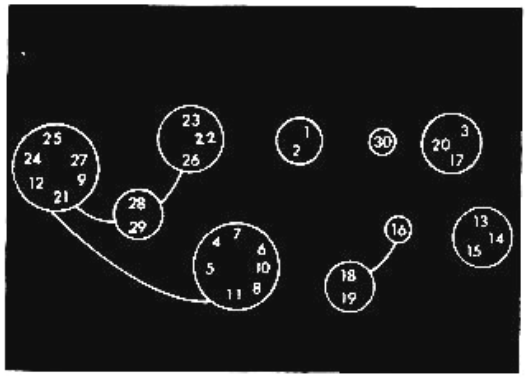



FIGURE 12—'NEW NUCLEI.' The figure shows the scene 'BRIDGE,' after LOCAL transforms it from the representation in Figure 11 'NUCLEI' to the one shown here.

SINGLEBODY. A strong link joining a nucleus and another nucleus composed by a single region is considered enough evidence and the nuclei in question merged, if there is no other link emanating from the single region. For instance, in Figure 12 'NEW NUCLEI,' nucleus :16 is merged with nucleus :18—:19 (see Figure 13 'FINAL'). Nucleus :28 - 29 is not joined with :26—22—23 or with :24—25—27—12—21—9. Even if nucleus :28—29 were composed by a single region, it still will not be merged, since two links emerged from it: two nuclei claim its possession.

RESULTS. After having screened out the regions that belong to the background, the nuclei are printed as "bodies."

In this process, the links which may be joining some of the nuclei are ignored: RESULTS considers the links of Figure 13 'FINAL', for instance, as non-existent.

## Summary

SEE uses a variety of kinds of evidence to "link" together regions of a scene. The links in SEE are supposed to be general enough to make SEE an object-analysis system. Each link is a piece of evidence that suggests that two or more regions come from the same object, and regions that get tied together by enough evidence are considered as "nuclei" of possible objects.

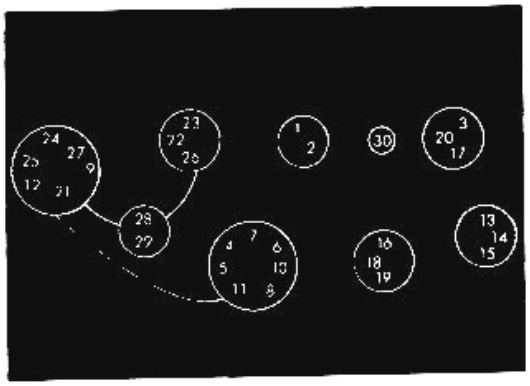


FIGURE 13—'FINAL.' SINGLEBODY joints the lonely region :16 with the nucleus :18–19.

*Some interesting examples*

We present in this section several scenes with the results obtained by SEE.

Example. **'CORN.'** The program analyzes the scene CORN (see figure 14 'CORN') obtaining the following identification of bodies:

(SINGLEBODY ASSUMES (:2 :3 :1) (:4) SAME BODY)
(BODY 1. IS :2 :3 :1 :4)
(BODY 2. IS :7 :6 :5)
(BODY 3. IS :13 :11 :12)
(BODY 4. IS :15 :16 :14) Results for 'CORN'
(BODY 5. IS :19 :18 :17)
(BODY 6. IS :21 :20)
(BODY 7. IS :8 :9 :10)

The region :4 got a single link with :3, and SINGLEBODY had to join it with the nucleus :1 :2 :3. Note that :4 and :12 did not get joined. The pyramid at the top was easily identified because its peak produces many links.

Example. **'MOMO'.** (See Figure 15 'MOMO'). The results are as follows:

(LOCAL ASSUMES (:17) (:9) SAME BODY)
(LOCAL ASSUMES (:9 :17) (:18) SAME BODY)
(BODY 1. IS :3 :2 :1)
(BODY 2. IS :32 :33 :27 :26)
(BODY 3. IS :28 :31)
(BODY 4. IS :19 :20 :34 :30 :29)
(BODY 5. IS :36 :35)
(BODY 6. IS :24 :5 :21 :4)
(BODY 7. IS :25 :23 :22) Results for 'MOMO'
(BODY 8. IS :14 :13 :15)
(BODY 9. IS :10 :16 :11 :12)
(BODY 10. IS :18 :9 :17)
(BODY 11 IS :7 :8)
(BODY 12. IS :38 :37 :39)

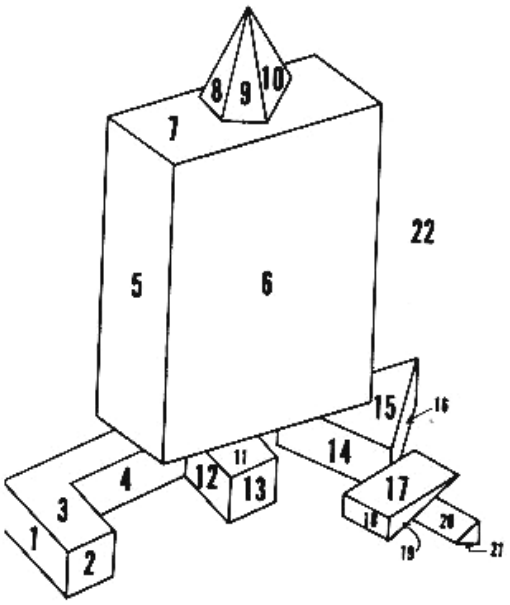


FIGURE 14—'CORN.' Since a link between :4 and :12 is not established, the bodies found in that part of the scene are :1 :2 :3 :4 and :11 :12 :13.

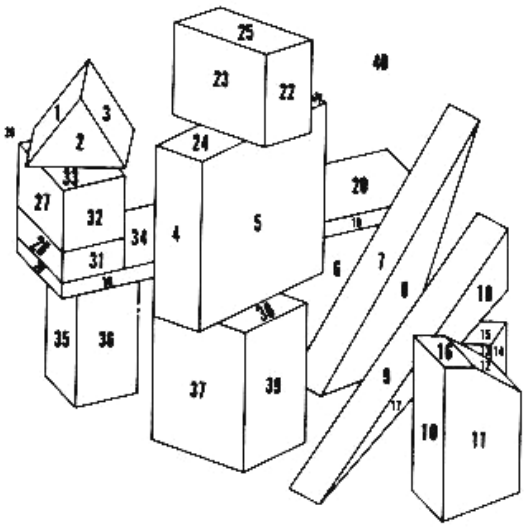


FIGURE 15—'MOMO.'

*Comments on the solution for 'MOMO':* The central cubes are easily isolated. :21 gets a link with :5 and another with :24, so there is no need to use LOCAL in it. The same is true of :26 with :27 and :33.

There is enough strong evidence to joint :28 and :31.

The links for :29 :30 :34 :20 :19 and :9 :17 :18 are indicated in Figure 16 'MOMO-LINKS.'

The dotted links between regions :30 and :34, and between :20 and :19 (see Figure 15 'MOMO') are due to the heuristic of parallel lines (of Figure 8 'PARALLEL'). These links made it possible to join the otherwise disconnected nuclei :29- :30- :19 and :34- :20. In particular, if :34 or :20 did not have parallel sides, SEE would have failed to merge these nuclei, and would have reported two bodies. The disposition of regions in MOMO is such that no link is present between :29 and :34, since :28 and :31 fall "exactly" parallel to :29 and :30. In a less extreme scene, some of these links would be present, allowing correct identification even if :29-:30-:34-:20-:19 were not a prism. Anyway, SEE did not make any mistakes.

The triangle of links formed by :9, :18 and :17 normally would have produced 3 separate bodies (since we need at least one double link to merge two regions); nevertheless, in this case, LOCAL makes some assumptions and saves the situation. Again, if :9 were not a parallelogram, there would be no weak links between :9 and :18 or between :9 and :17, and 3 bodies would have been reported instead of :9-:17-:18. But we should notice that, in general, two links instead of one would be between :17 and :18.

Links were extablished between :12 and :13, without serious consequences, because we need two mistakes, two wrong strong links, to fool the program. But that could happen.

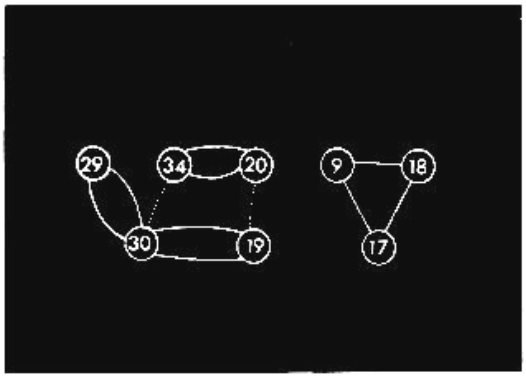



FIGURE 16—MOMO—LINKS'. Some links of figure 15 'MOMO'

Example. **'HOME'**.—(see scene HOME). The results are in Table V 'RESULTS FOR HOME.'

(SINGLEBODY ASSUMES (:38) (:39) SAME BODY)
(SINGLEBODY ASSUMES (:34) (:33) SAME BODY)
(SINGLEBODY ASSUMES (:25 :24 :22) (:23) SAME BODY)
(SINGLEBODY ASSUMES (:20) (:21) SAME BODY)
RESULTS
(BODY 1. IS :3 :6 :10 :2)
(BODY 2. IS :14 :13 :11 :12)
(BODY 3. IS :0 :8)
(BODY 4. IS :18)
(BODY 5. IS :15 :5)
(BODY 6. IS :38 :39)
(BODY 7. IS :7 :26 :27)
(BODY 8. IS :20 :21)
(BODY 9. IS :29 :28)
(BODY 10. IS :34 :33)
(BODY 11. IS :31 :32 :30)
(BODY 12. IS :25 :24 :22 :23)
(BODY 13. IS :16)
(BODY 14. IS :19)
(BODY 15. IS :17)
(BODY 16. IS :36 :37 :35)

Results for HOME

TABLE V—Results for HOME

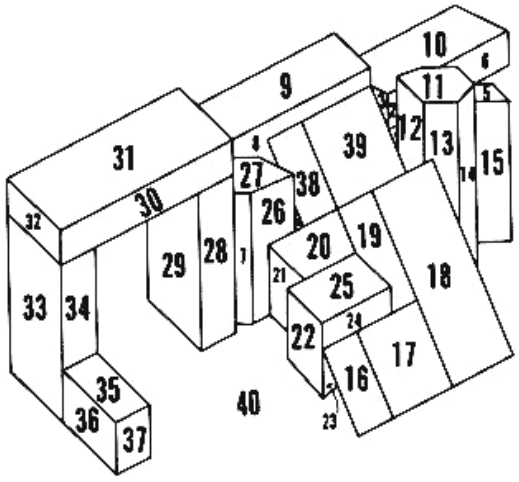



FIGURE 17—'HOME.' The hexagonal prisms did not create difficulties; SINGLEBODY was needed to group :34 with :33. The body :38–39 was identified as a single one, but :16–17 was reported as two. Note that there does not exist local link between :21 and :20; nevertheless, SINGLEBODY makes the correct identification. (See Table V).

*Comments on the solution to HOME:* There is a certain degree of ambiguity in this scene which human beings tend to resolve by assuming parallelepipeds. :16 and :17

are reported as two separate bodies, instead of one, which is probably a more natural answer.

In the picture from which 'HOME' was drawn, :19 and :18 were the two faces of a single parallelepiped leaning on :38-:39; SEE reports :19 as one body and :18 as another.

Nevertheless, SEE reports :38 and :39 as forming part of the same body (which was in fact the case in the picture in question), due to the fact that :4 and :1 are background.

Example. **'SPREAD.'**—(see Figure 18 'SPREAD'). The results are in Table VI 'RESULTS FOR SPREAD'.

(LOCAL ASSUMES (:36) (:4) SAME BODY)
(LOCAL ASSUMES (:30) (:31 :32 :29) SAME BODY
(LOCAL ASSUMES (:16) (:17) SAME BODY)
(LOCAL ASSUMES (:5) (:20) SAME BODY)
(LOCAL ASSUMES (:3 :11 :9) (:12 :10) SAME BODY)
(SINGLEBODY ASSUMES (:23) (:22) SAME BODY)
(SINGLEBODY ASSUMES (:4 :36) (:37) SAME BODY)
(SINGLEBODY ASSUMES (28 :26 :27) (:25) SAME BODY)
RESULTS
(BODY 1. IS :39 :40 :38)
(BODY 2. IS :23 :22)
(BODY 3. IS :42 :41)
(BODY 4. IS :4 :36 :37)
(BODY 5. IS :24)
(BODY 6. IS :28 :26 :27 :25) Results for SPREAD
(BODY 7. IS :31 :32 :29 :30)
(BODY 8. IS :20 :5)
(BODY 9. IS :12 :10 :3 :11 :9)
(BODY 10. IS :13 :7 :1)
(BODY 11. IS :21 :6)
(BODY 12. IS :8 :18)
(BODY 13. IS :17 :16)
(BODY 14. IS :45 :43 :44)
(BODY 15. IS :19)
(BODY 16. IS :15 :14)

TABLE VI—Results for SPREAD

*Comments on the solution to SPREAD*: The body :22-23-24, due to insufficiency of links, was split in two: :22-23 and :24.

Since there is only one link between :6 and :5, this body gets split into two: :6-21 and :5-20. Note that :21 is *not* the same face as :20, and there is where SEE gets confused and refuses to see evidence toward linking :21 with :20.

The long body :9-10-11-12-3 gets properly identified.

Example. **'HARD.'**—This scene is analyzed with the following results:

(LOCAL ASSUMES (:11) (:12) SAME BODY)
(LOCAL ASSUMES (:15) (:16) SAME BODY)
RESULTS
(BODY 1. IS :12 :11)
(BODY 2. IS :16 :15)
(BODY 3. IS :32 :31 :30)
(BODY 4. IS :9 :10 :8)
(BODY 5. IS :18 :19 :20)
(BODY 6. IS :13 :17 :14) Results for 'HARD'
(BODY 7. IS :5 :4)
(BODY 8. IS :1 :2 :33)
(BODY 9. IS :24 :23 :22 :3 :21 :28 :29)
(BODY 10. IS :25 :26 :27)
(BODY 11. IS :7)
(BODY 12. IS :6)

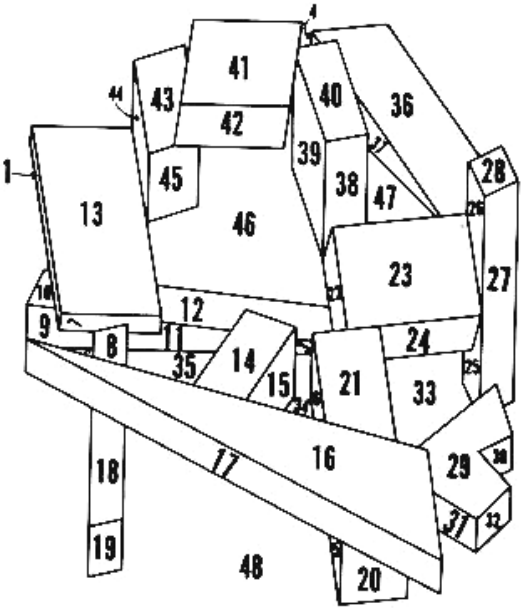



FIGURE 18—'SPREAD.' :41 and :42 are identified as a single body. Nevertheless, :8-18-19 gests broken into :8-18 and :19. :28-27-26-25 gets correctly identified, as well as the funny looking :29-30-31-32. (See Table VI).

*Comments on the solution to HARD:* :15 and :16 have to make use of weak evidence to get joined and recognized as forming part of the same body. Nevertheless, tois is not necessary with :28 and :29, because, through a chain of Ts, there is enough evidence in :3, :21 and :22 to join successfully all that long and twice occluded body.

There is one serious bug in this identification: regions :7 and :6 get identified as two separate bodies, and not as a single one, as one would normally do. This is caused by the fact that neither :7 nor :6 have visible 'useful' vertices, and there are not enough parallel lines in them to use the heuristic of Figure 8 'PARALLEL.'

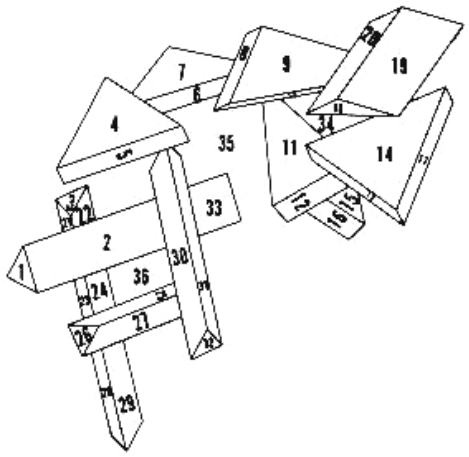



FIGURE 19—'HARD.' Body :21–22–3–23–24–28–29 is reported as a single object, which is correct. Nevertheless, regions :7 and :6 get reported as two bodies.

:33 was recognized as part of :1-2, as it should be.

## DISCUSSION

We have described a program that analyzes a three-dimensional scene (presented in the form of a line drawing) and splits it into "objects" on the basis of pure form. If we consider a scene as a set of regions (surfaces), then SEE partitions the set into appropriate subsets, each subset forming a three-dimensional body or object.

The performance of SEE shows to us that *it is possible to separate a scene into the objects forming it, without needing to know in detail these objects;* SEE does not need to know the 'definitions' or descriptions of a pyramid, or a pentagonal prism, in order to isolate these objects in a scene containing them, even in the case where they are partially occluded.

The basic idea behind SEE is to make global use of information collected locally at each vertex: this information is noisy and SEE has ways to combine many different kinds of unreliable evidence to make fairly reliable global judgments.

The essentials are:

(1) Representation as vertices (with coordinates), lines and regions
(2) Types of vertices.
(3) Concepts of links (strong and weak), nuclei and rules for forming them.

The current version of SEE is restricted to scenes presented in symbolic form. It does not have resources for dealing with curves, shadows, noise, missing lines, etc. So it represents a "geometric theory of object identity" at present.

Since SEE requires two strong evidences to join two nuclei, it appears that its judgments will lie in the 'safe' side, that is, SEE will almost never join two regions that belong to different bodies. From the analysis of scenes shown above, its errors are almost always of the same type: regions that should be joined are left separated. We could say that SEE behaves "conservatively," especially in the presence of ambiguities.

Divisions of the evidence into two types, strong and weak, results in a good compromise. The weak evidence is considered to favor linking the regions, but this evidence is used only to reinforce evidence from more reliable clues. Indeed, the weak links that give extra weight to nearly parallel lines are a concession to object-recognition, in the sense of letting the analysis system exploit the fact that rectangular objects are common enough in the real world to warrant special attention.

Most of the ideas in SEE will work on curves too. However, we do not yet have a good idea of how sensitive the system will be to "symbolic noise," i.e., missing misplaced, and false region boundaries. As indicated in the scenes above, the system seems to have good resistance to "accidents" in which objects happen to to "line up" unusually well. This feature may be necessary if the preprocessor that (eventually) will feed data SEE decides to report two nearly colinear lines as one alone, or if it lumps several vertices into one, because they lie close to each other.

*Extensions.* (None of this incorporated in the actual program.) More heuristics could be added to increase the number of links; in particular, given a good number of "link proposers," parameters set outside (by the user) would tell SEE which set of heuristics to use; for instance, if we knew that the scene is formed by prisms, we could use the heuristics that ask for parallel lines having a given configuration, we could check the length of certain edges, etc.

Different kinds of links could also be established; in this way, 'contradictory' links (such as the three links of Figure 13 'FINAL' which SEE just ignores) could be studied further, in order to discover ambiguities. In particular, a "conditional link" would be useful: regions :2 and :3 belong to the same body if region :4 does not.

SINGLEBODY could be improved so as to analyze in more detail the regions that by themselves form bodies (that is, the bodies formed by only one region); in this way, we could hope to joint regions :6 and :7 of scene 'HARD.'

## ACKNOWLEDGMENTS

The author is grateful to Professors Marvin Minsky and Seymour Papert for several helpful discussions, and to the referees of this paper for their constructive recommendations.

Michael Speciner made valuable suggestions; Cornelia Sullivan helped to codify most of the scenes.

"The author was partially supported by a scholarship from the Instituto Nacional de la Investigación Científica (México)."

The work reported here could not have been produced had the author not had access to the computational facilities of Project MAC, an M.I.T. research program sponsored by the Advanced Research Projects Agency, Department of Defense, under Office of Naval Research Contract Number Nonr-4102(01). This paper, having had this support from MAC, may be reproduced in whole or in part for any purpose of the United States Government.

## REFERENCES

1 R GREENBLATT J HOLLOWAY
*Sides 21*
Memorandum MAC–M–320 A I Memo 101 Project MAC MIT August 1966 A program that finds lines using gradient measurements

2 K K PINGLE
*A program to find objects in a picture*
Memorandum No 39 Artificial Intelligence Project Stanford University January 1966 It traces around objects in a picture and fits curves to the edges

3 A GUZMÁN
*Scene analysis using the concept of model*
Technical Report No AFCRL–67–0133 Computer Corporation of America Cambridge Massachusetts January 1967

4 A GUZMÁN
*A primitive recognizer of figures in a scene*
Memorandum MAC–M–342 AI Memo 119 Project MAC MIT January 1967

5 A GUZMÁN
*Some aspects of pattern recognition by computer*
MS Thesis Electrical Engineering Department MIT February 1967 Also available as a Project MAC Technical Report MAC–TR–37 This memorandum discusses many methods of identifying objects of known forms.

6 A GUZMÁN H V McINTOSH
*CONVERT*
Communications of the ACM
9,8 August 1966 pp 604–615 Also available as Project MAC Memorandum MAC–M–316 AI Memo 99 June 1966

7 A GUZMÁN
*Decomposition of a visual scene into bodies*
Memorandum MAC–M–357 AI Memo 139 Project MAC MIT September 1967 unpublished This memorandum is a more technical description of SEE

8 R H CANADAY
*The description of overlapping figures*
MS Thesis Electrical Engineering Department MIT January 1962

9 L G ROBERTS
*Machine perception of three-dimensional solids*
Optical and electrooptical information processing pp 159–197 J T Tippett et al eds MIT Press 1965

10 M L MINSKY S A PAPERT
*Research on intelligent automata*
Status report II
Project MAC MIT September 1967

11 J McCARTHY
*Plans for the Stanford artificial intelligence project*
Stanford University April 1965

12 C A ROSEN N J NILSSON eds
*Application of intelligent automata to reconnaissance*
Third Interim Report Stanford Research Institute December 1967