MASSACHUSETTS INSTITUTE OF TECHNOLOGY
PROJECT MAC

Artificial Intelligence
Memo No. 192 August 1970

# REMOVING SHADOWS IN A SCENE

Richard Orban

This paper describes a LISP function, ERASER, to be used in the process of recognizing objects by computer. It is a pre-processor to a program called SEE which finds whole bodies in scenes. A short description of SEE and the required data-form for a scene is given. SEE is simulated for five different scenes to demonstrate the effects of shadows on its operation. The function ERASER is explained through its sequence of operation, the heuristics used and detailed results for test cases. Finally, a "how to use it" section describes the data required to be on the property lists of the vertices in the scene, and the cruft that ERASER puts on these p-lists as it operates.

Work reported herein was supported by the Warren McCulloch Laboratory, an M.I.T. research program sponsored by the Advanced Research Projects Agency of the Department of Defense under Office of Naval Research contract number N00014-70-A-0362-0002.

Reproduction of this document, in whole or in part, is permitted for any purpose of the United States Government.

## Introduction

This paper describes a LISP function, ERASER, used as an aid in the interpreting of a real-life scene. The scenes considered are random groupings of rectangular blocks of various sizes. A representation of such a scene, useful for manipulation, is in the form of a network of links, showing how, when viewed from some angle, the vertices of the solid objects are connected. The network is a mapping of the three-dimensional scene into a two-dimensional representation and would be derived previously, by an exterior program, from vidisector data.

ERASER is used to remove vertices and links caused by shadows, from the network. With the superfluous data removed, the network can be analyzed more successfully by SEE, a function that finds bodies in scenes.

## Background Information

To allow a proper perspective for understanding what ERASER does and how it operates, a short description of another program, which handles the same data-type, is necessary. For a more detailed discussion, see references (1) and (2).

The present process used for the recognition of objects by vision encompasses three steps:

1. A program to convert the light intensity information of a visual scene, obtained from a vidisector, into a network of vertices. In a LISP representation, the network is a list, whose elements provide information about each vertex visible to the vidisector. This information would be: the name of the vertex, the x- and y-coordinates, and the neighboring vertices (i.e. those vertices in the network connected to the named vertex). For example, a cube might be:

```
(  A (1 1) (F B)
   B (1 6) (A E C)
   C (4 8) (B D)
   D (9 8) (G C E)
   E (6 6) (F D B)
   F (6 1) (A G E)
   G (9 3) (F D)    )
```

Figure 1a is a more natural, equivalent way of representing this same cube.

2. A program for processing the input data and determining whole bodies.

3. A matching program to classify the bodies as well-understood objects -- for which the system has convenient and easy to manipulate models.

The second step is accomplished by SEE, a Ph.D. thesis by Adolfo Guzman, in conjunction with a pre-processor. The pre-processor operates on the vertex network and derives information necessary for

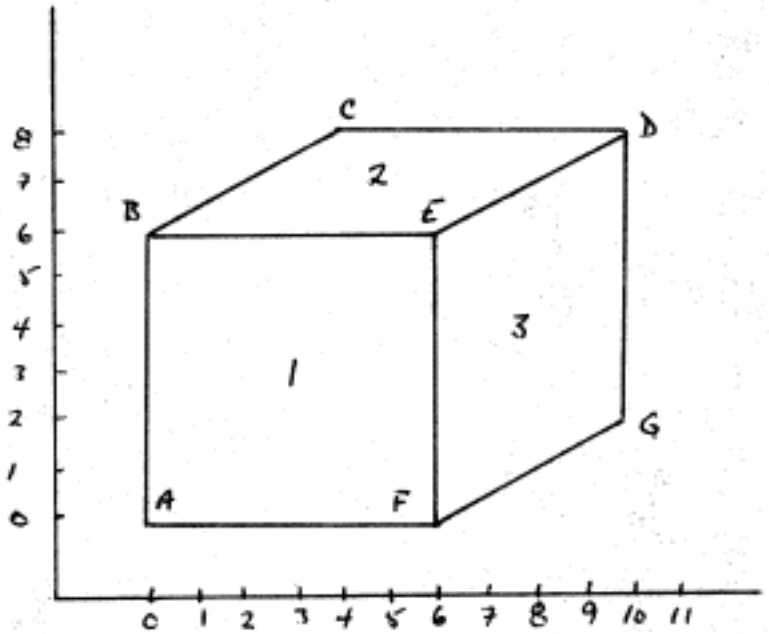



Figure 1a

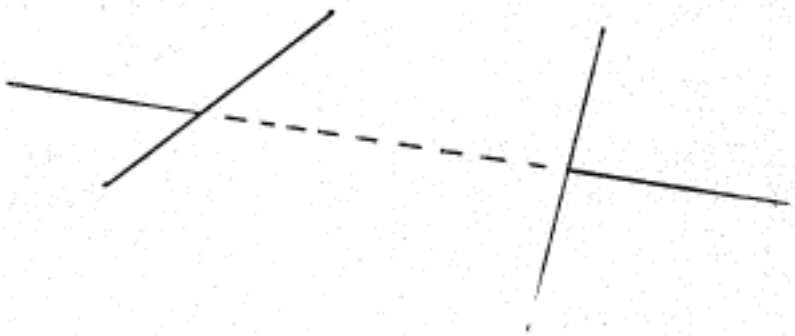
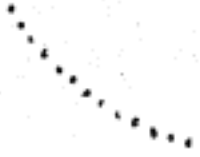
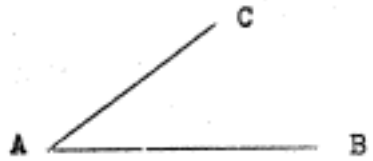
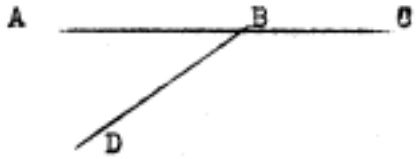


Figure 1b

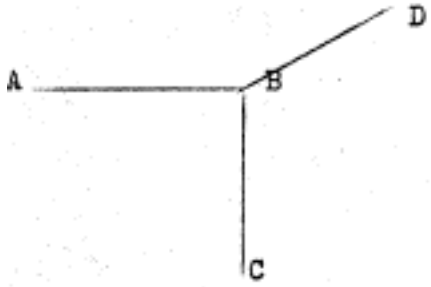




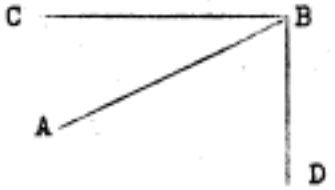

L



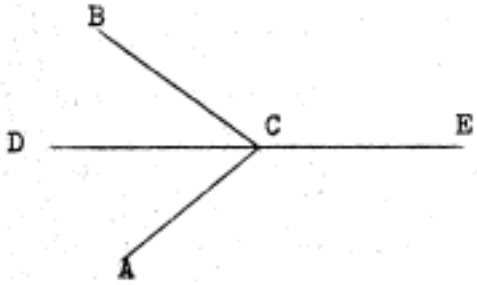

T



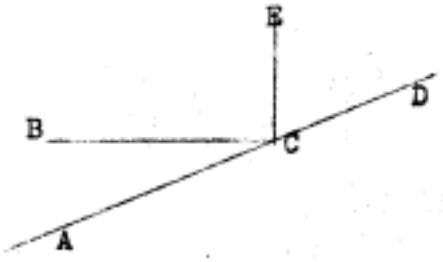

FORK



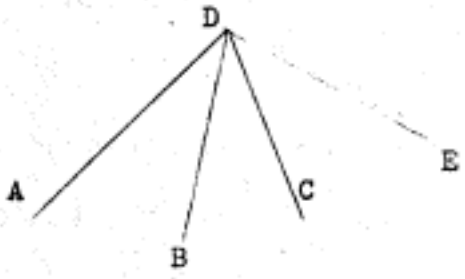

ARROW



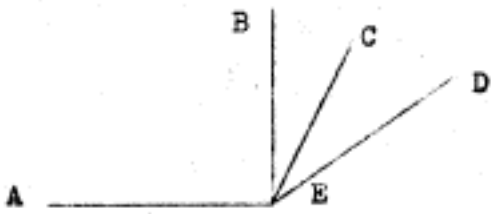

X




K




PEAK




MULTI

Figure 2.

SEE. Regions are found and labelled. Each vertex is classified as being one of eight types (see Figure 2) -- L, T, X, K, ARROW, FORK, MULTI, or PEAK -- and information characteristic to that type is supplied. An L is a vertex linked to two other vertices. A vertex linked to three vertices can be a T, a FORK, or an ARROW. The vertex with four neighboring vertices is either an X or a K. A vertex not fitting one of the above categories is a MULTI, or if the y-coordinate is greater than that of any of the linking vertices the special case PEAK is used. For Figure 1:

Regions: 1, 2, 3

Vertices of type L: A, C, G

Vertices of type FORK: E

Vertices of type ARROW: B, D, F

Operating on the network and the information supplied by the preprocessor, SEE makes reasonable guesses concerning which regions form bodies. The actual process is to create a network of linked regions from the network of linked vertices. The new links, between the regions of the scene, are contructed from a systematic analysis of the vertices of the scene. Two degrees of association between the regions are possible. These are designated "strong" and "weak" links. Three functions in SEE merge regions to form nucleia according to the links that have been formed. GLOBAL requires that regions be connected by at least two strong links to be joined as a nucleus. LOCAL

joins regions having one strong link supported by a weak link. A single region having only one strong link, which connects it to only one nucleus, will be connected with that nucleus by SINGLEBODY. When all possible nucleia have been formed, any remaining links are ignored and SEE outputs each nucleus as being a body -- listing the regions comprising the bodies.

Except for recently explored and documented weaknesses or special problems (see reference (3)), SEE performs well in noise-free situations. The motivation for ERASER is to enable SEE to handle more complicated and realistic situations, namely those having shadows.

Allowing shadows, the heuristics used by SEE cause improper processing of the scene. This is understandable since these heuristics are local; specifically, observed characteristics of the transformations resulting from the mapping of three-space into two-space. A FORK is a front-on view of a corner; an ARROW is a side view of a corner. The T-type vertex can be either a degenerate view of an edge and corner or two regions of one body obscured by another distinct body, depending upon: the location of the background relative to the T, the existence of an "opposite" and "facing" T vertex (such a pair of matching Ts are shown in Figure 1b), or the slopes of segments bounding the regions that the T bounds. An X occurs as the interior edge of stacked blocks. In addition, some special case heuristics are used which help prevent links that would be formed due to optical illusions or point-of-view problems.

Although a heuristic may involve a search for a vertex quite removed from the vertex being analyzed, as in the case of the matching Ts, no explicit model or conjecture is made concerning what type of body is being pieced together. SEE's guesses about the structure of material bodies is based on from one to three edges of a single region.

With SEE in its present form, there are two obvious types of action to be expected for scenes having shadows:

1. In conservative situations, with insufficient links between real body regions and shadow regions, the shadow region is simply ignored and becomes a one-region body.

2. When links between shadow and body exist, the shadow is absorbed into the body and thereby complicates the body. It makes the job of the pattern-matching program more difficult.

## Simulation of SEE

SEE was simulated for five sample scenes of varying complexity, and the above-mentioned actions are observed to have occurred. The scenes are shown in Figures 3, 5, 7, 9, and 11. A diagram showing three successive stages in the operation of SEE follow the scenes (Figures 4, 6, 8, 10, and 12). Each such diagram shows all the links formed between the regions of the scene, the nucleia formed by the merging process, and the results as they might be printed out by SEE.

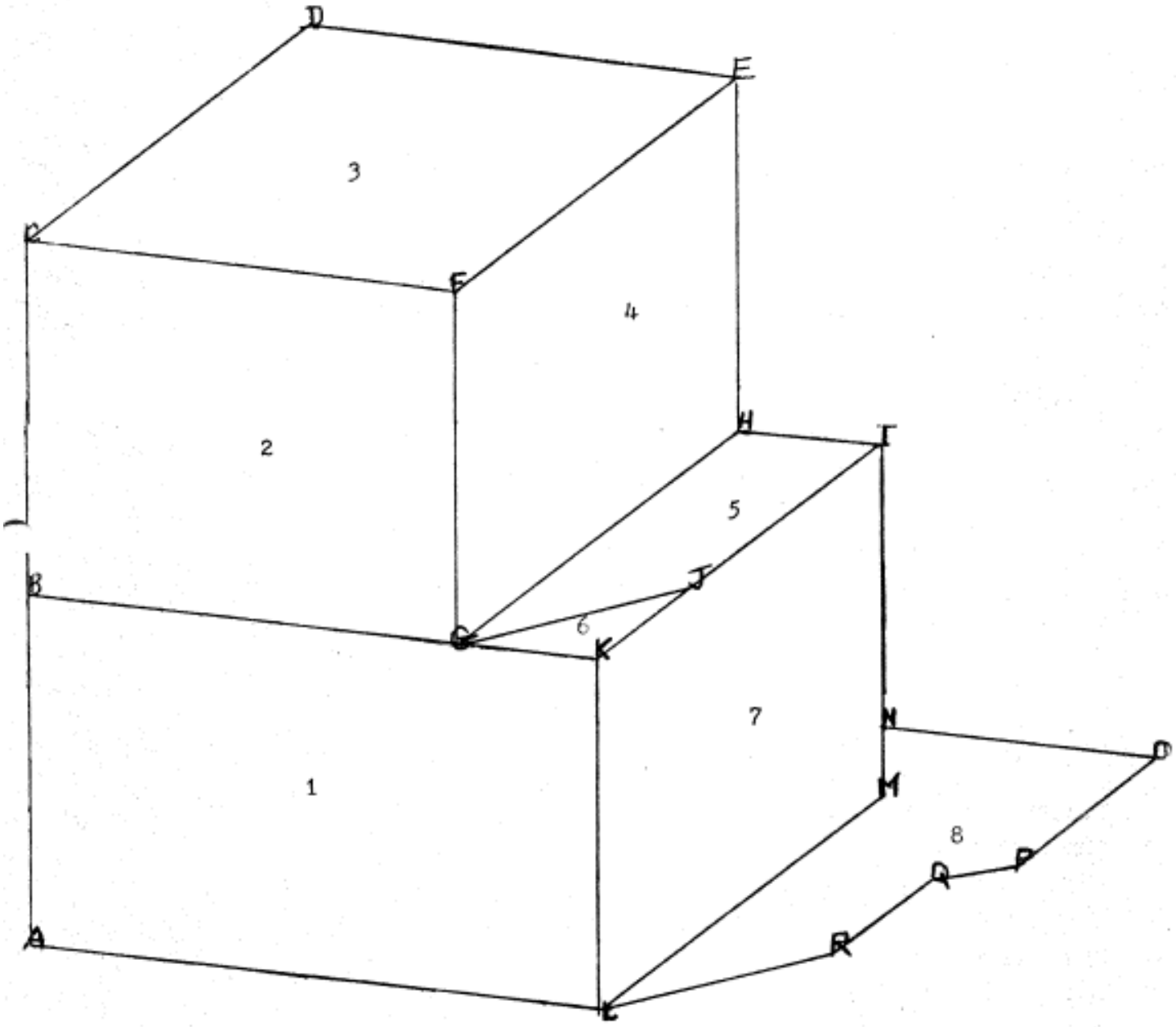
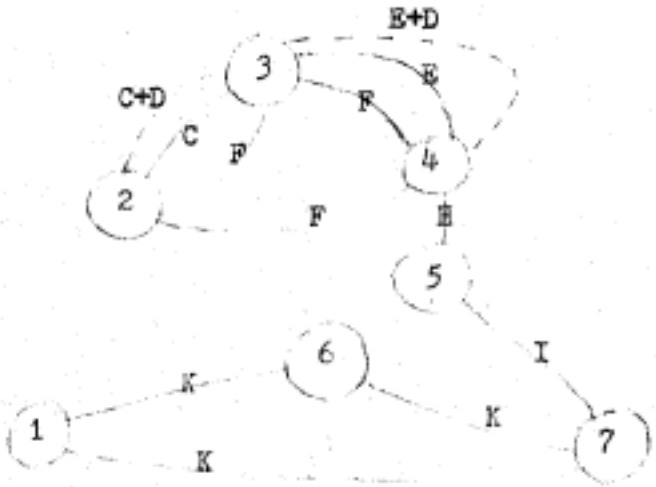
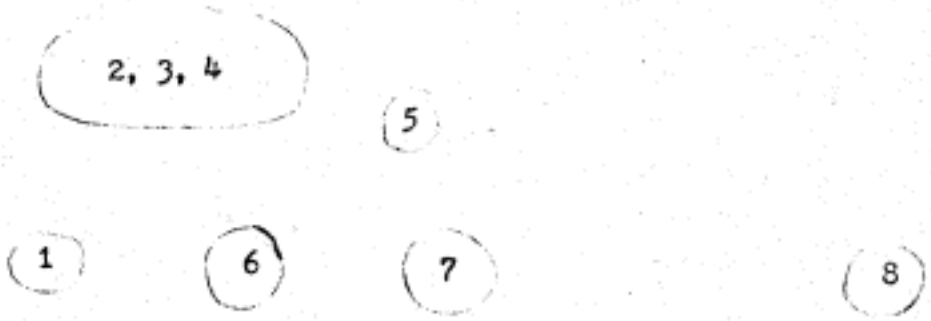



Figure 3.

LINKS




NUCLEIA




RESULTS

BODY 1 IS 2, 3, 4
BODY 2 IS 1
BODY 3 IS 6
BODY 4 IS 7
BODY 5 IS 5
BODY 6 IS 8

**Figure 4.**

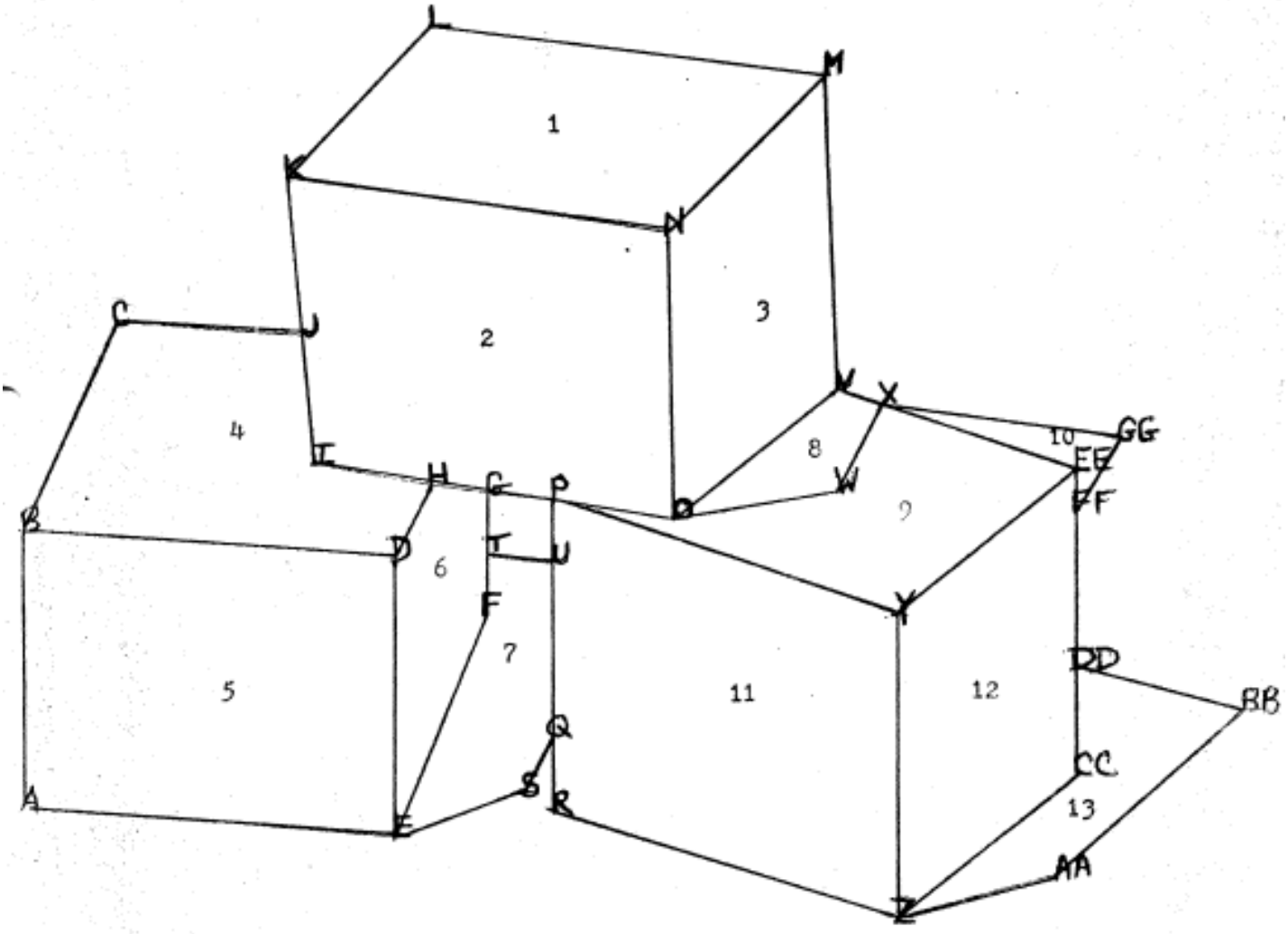
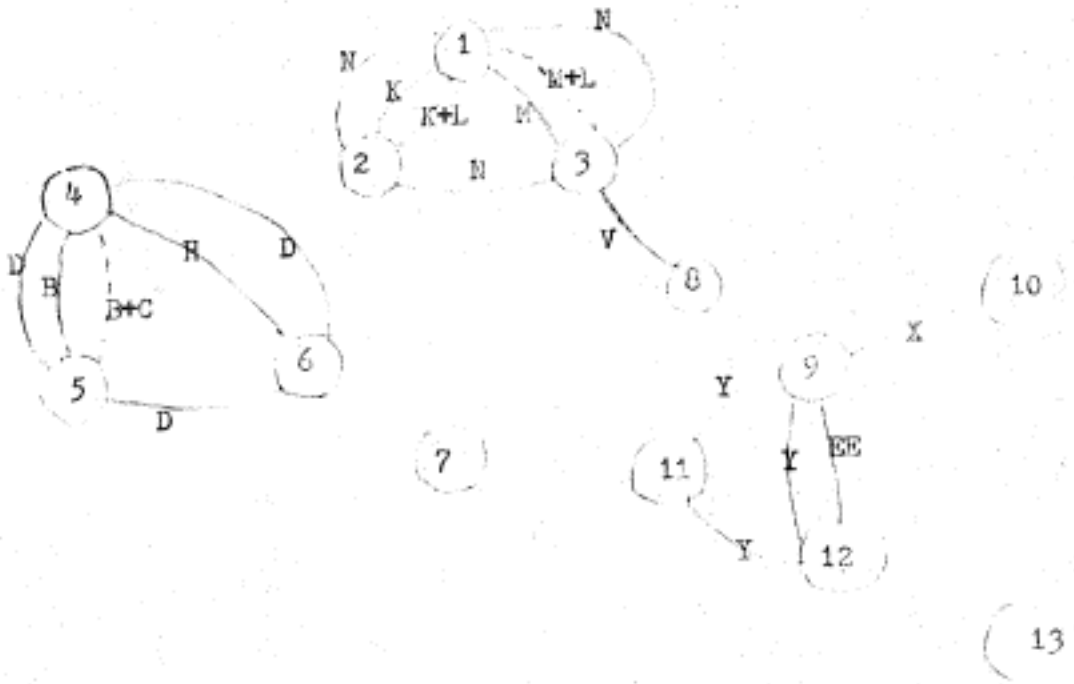

L
M
K
N
1
2
3
C
J
4
I
H
G
P
O
V
X
W
8
9
10
GG
EE
FF
B
D
6
T
U
F
7
Y
5
11
12
DD
BB
Q
S
R
CC
13
A
E
Z
AA

Figure 5.

LINKS




NUCLEIA

1,2,3,8

4,5,6

7

9, 10, 11, 12

13

RESULTS

BODY 1 IS 4, 5, 6
BODY 2 IS 1, 2, 3, 8
BODY 3 IS 7
BODY 4 IS 9, 10, 11, 12
BODY 5 IS 13

**Figure 6.**

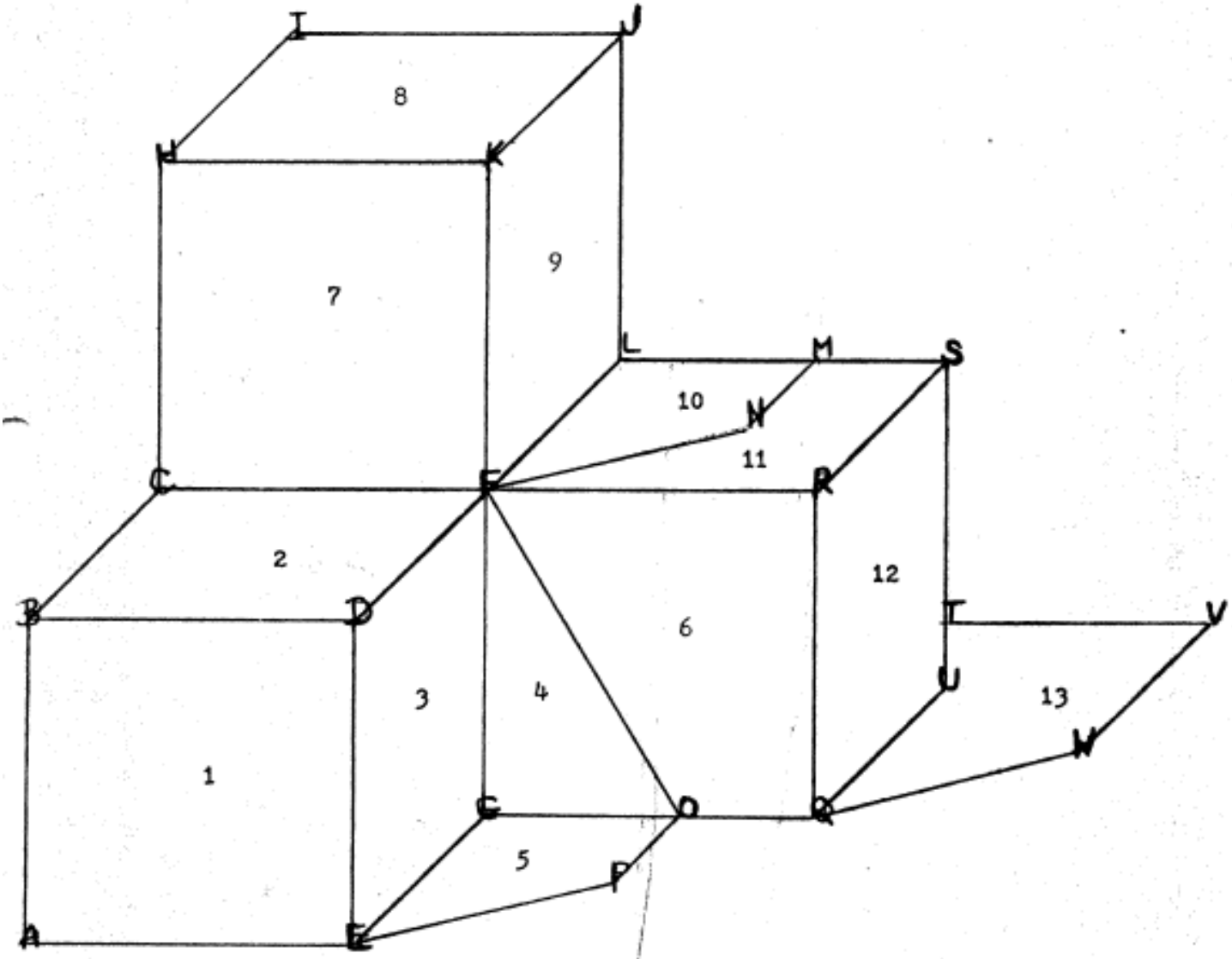
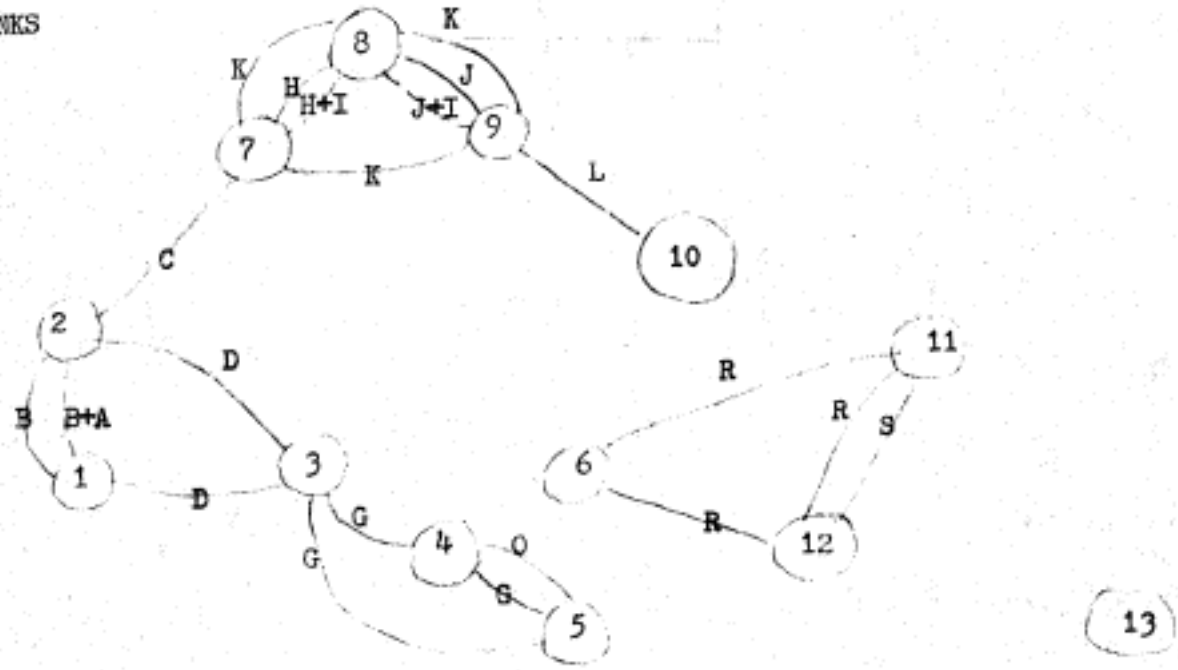
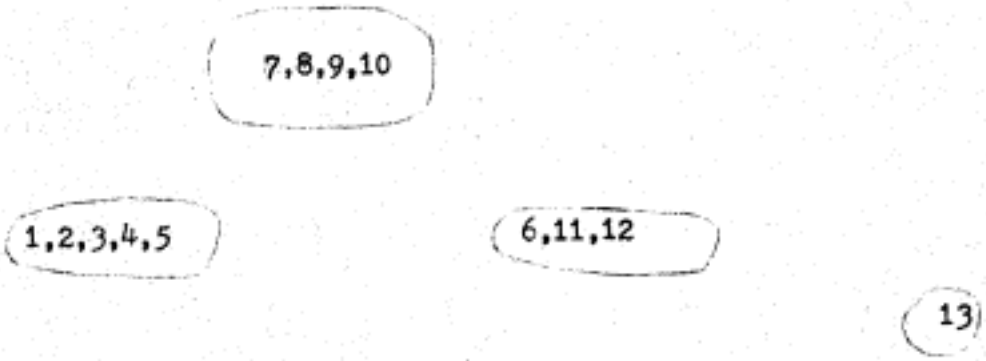



Figure 7.

LINKS




NUCLEIA




RESULTS

BODY 1 IS 1, 2, 3, 4, 5
BODY 2 IS 7, 8, 9, 10
BODY 3 IS 6, 11, 12
BODY 4 IS 13

Figure 8.

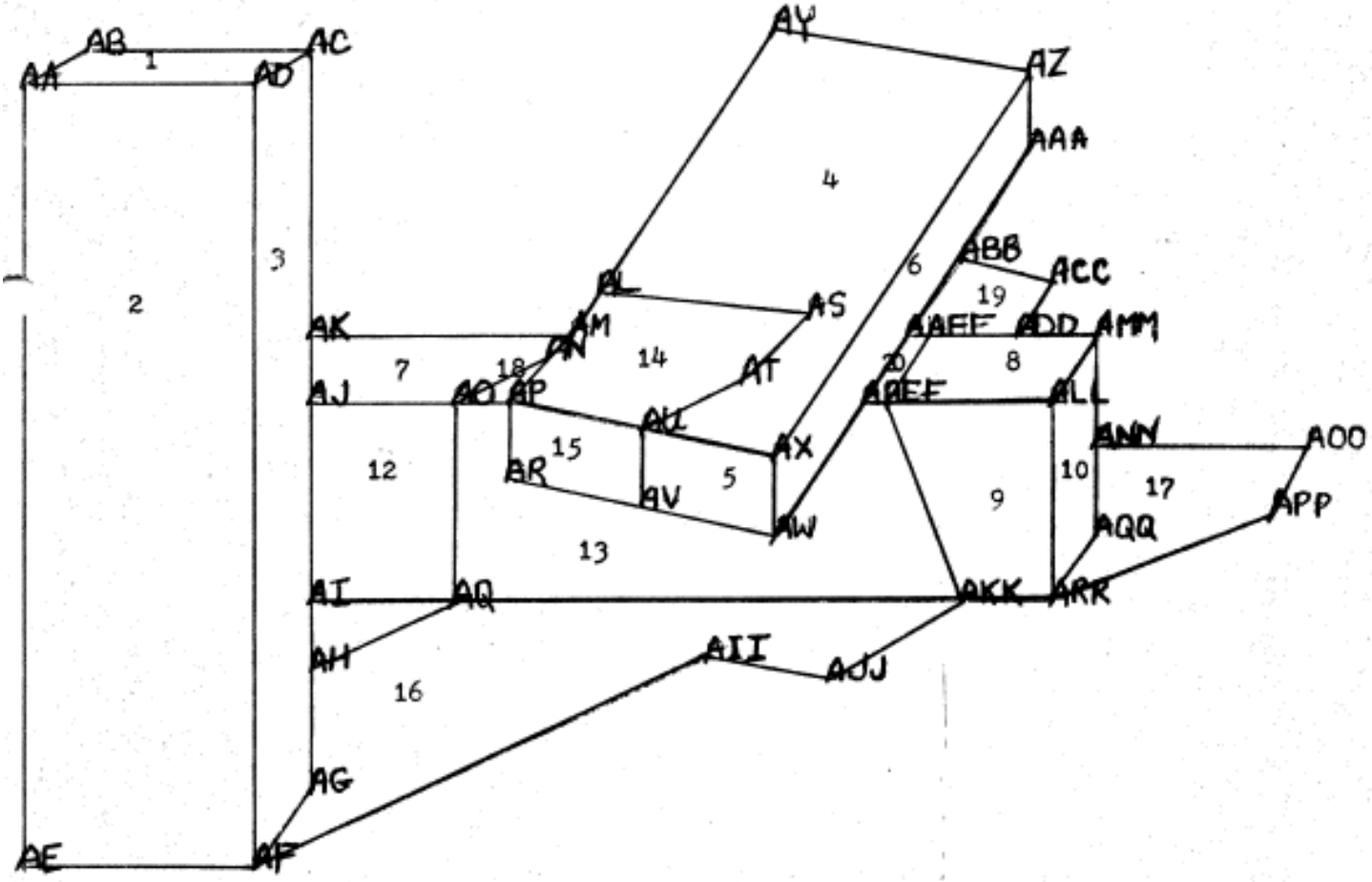
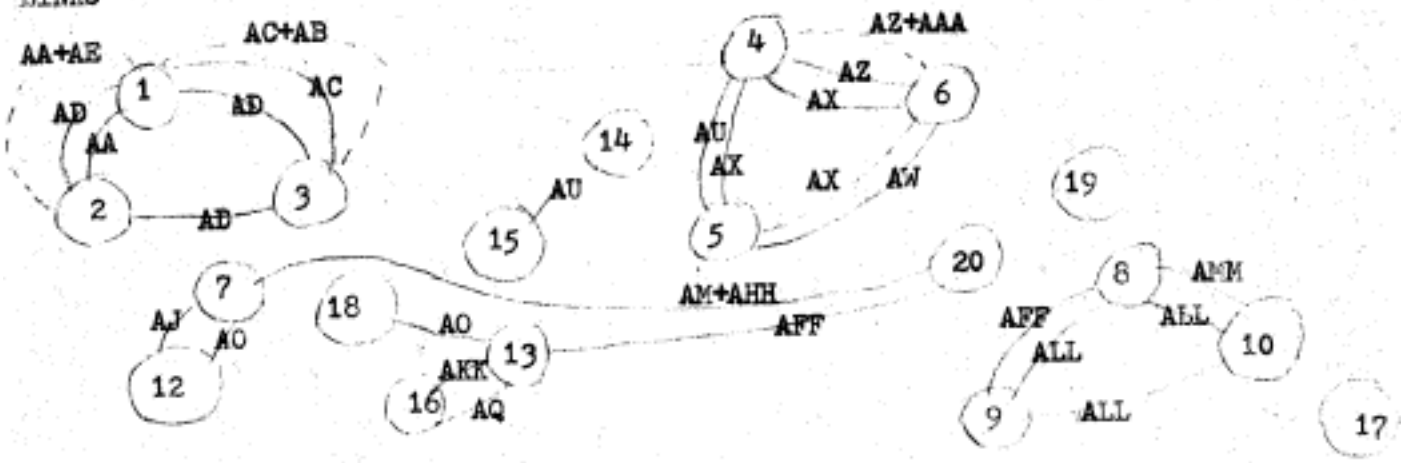
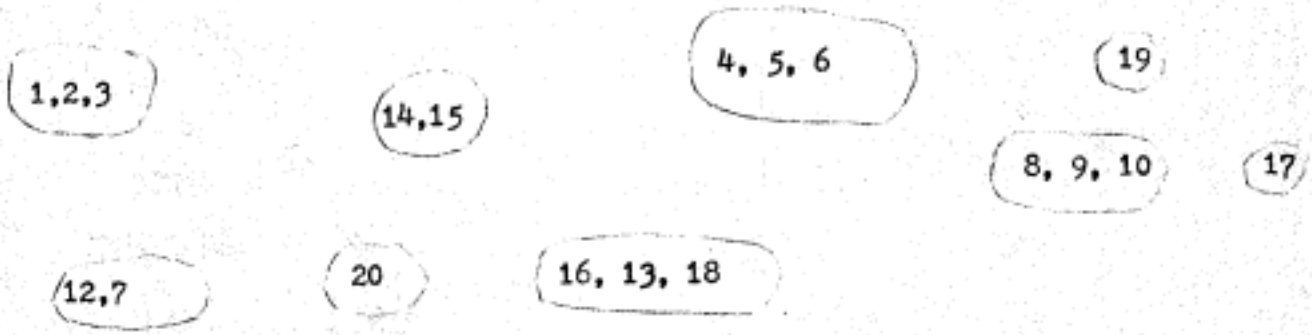

AA
AB
AC
AD
AE
AF
AG
AH
AI
AJ
AK
AL
AM
AN
AO
AP
AQ
AR
AS
AT
AU
AV
AW
AX
AY
AZ
AAA
ABB
ACC
ADD
AEE
AFF
AGG
AHH
AII
AJJ
AKK
ALL
AMM
ANN
AOO
APP
AQQ
ARR
1
2
3
4
5
6
7
8
9
10
12
13
14
15
16
17
18
19
20

Figure 9.

LINKS




NUCLEIA




RESULTS

BODY 1 IS 1, 2, 3,
BODY 2 IS 14, 15
BODY 3 IS 4, 5, 6
BODY 4 IS 19
BODY 5 IS 8, 9, 10
BODY 6 IS 12, 7
BODY 7 IS 17
BODY 8 IS 20
BODY 9 IS 16, 13, 18

Figure 10.

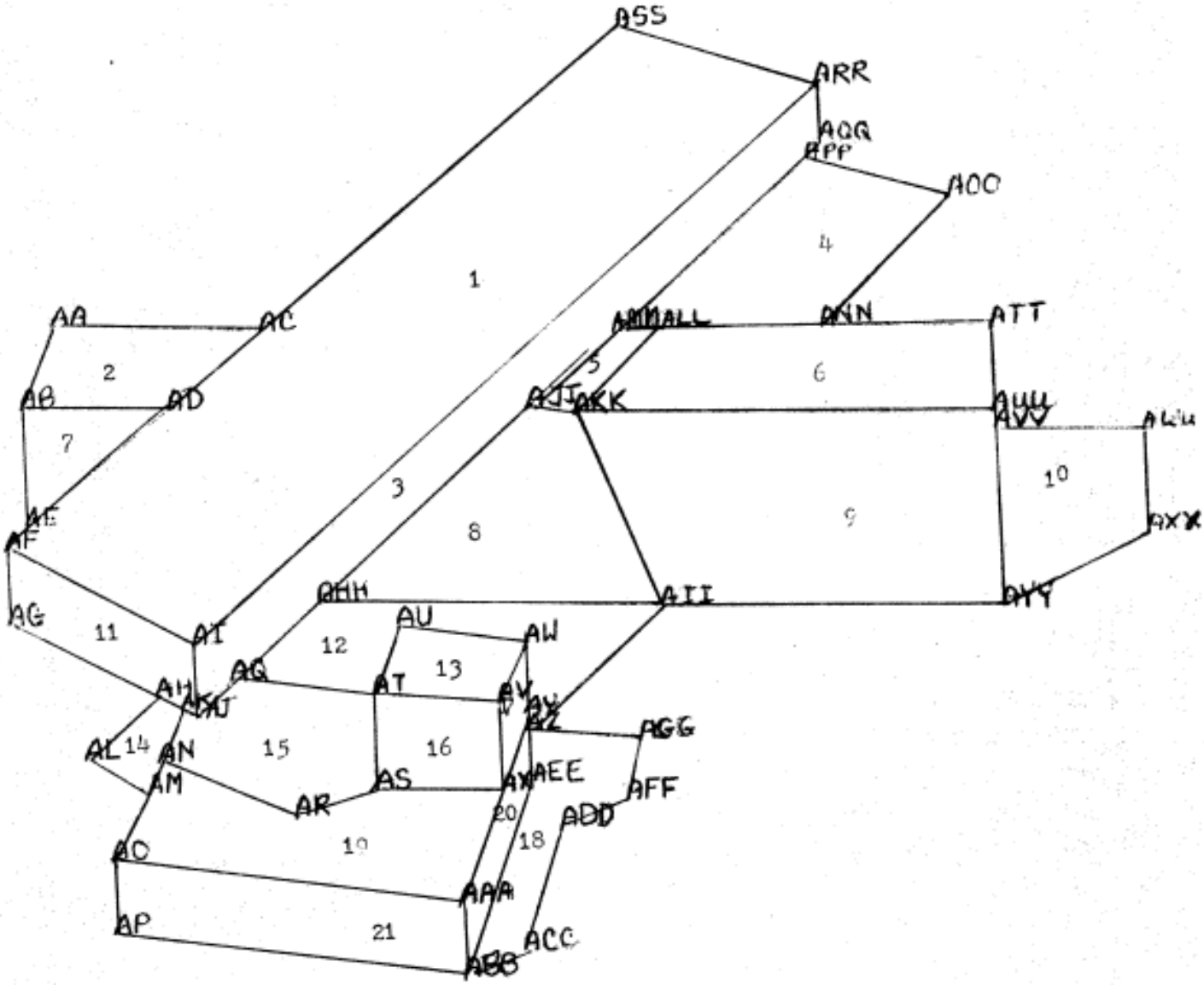
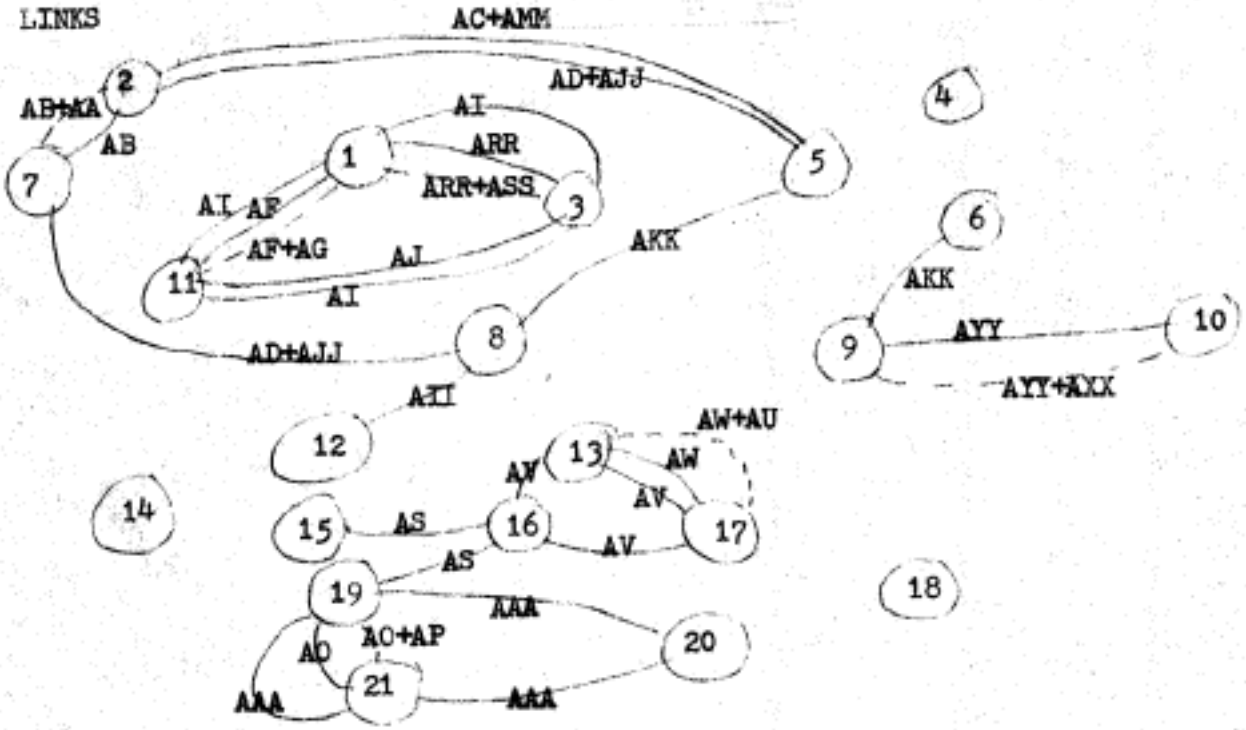
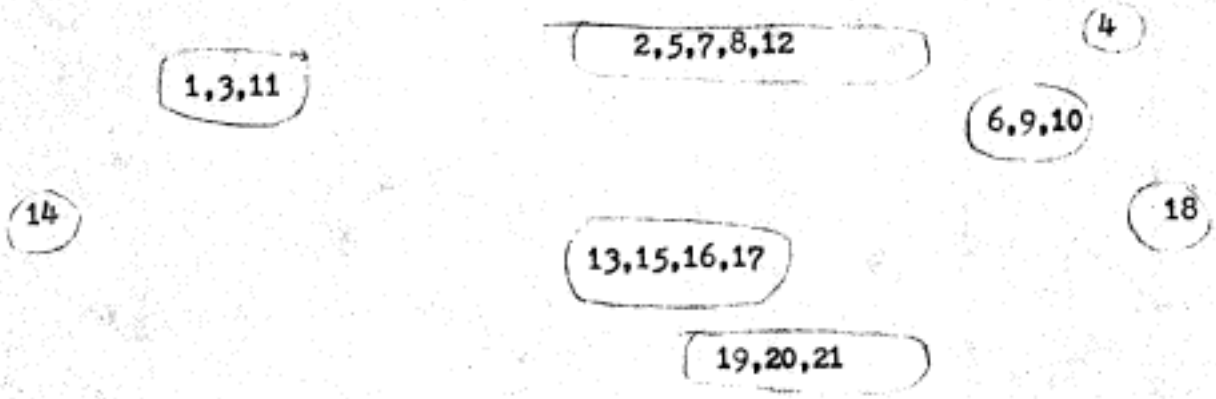

ASS
ARR
AQQ
APP
AOO
1
4
AA
AC
ALL
ANN
ATT
2
5
6
AB
AD
AKK
AUU
AVV
AWW
7
3
10
AE
8
9
AXX
AF
AHH
AII
AYY
AG
11
AI
AU
AW
12
13
AQ
AT
AH
AV
AZ
AGG
AL
14
AN
15
16
AM
AS
AEE
AFF
AR
20
ADD
18
19
AO
AAA
AP
21
ACC
ABB

Figure 11.




NUCLEIA




RESULTS

BODY 1 IS 14
BODY 2 IS 1, 3, 11
BODY 3 IS 2, 5, 7, 8, 12
BODY 4 IS 13, 16, 17, 15
BODY 5 IS 19, 20, 21
BODY 6 IS 4
BODY 7 IS 6, 9, 10
BODY 8 IS 18

Figure 12.

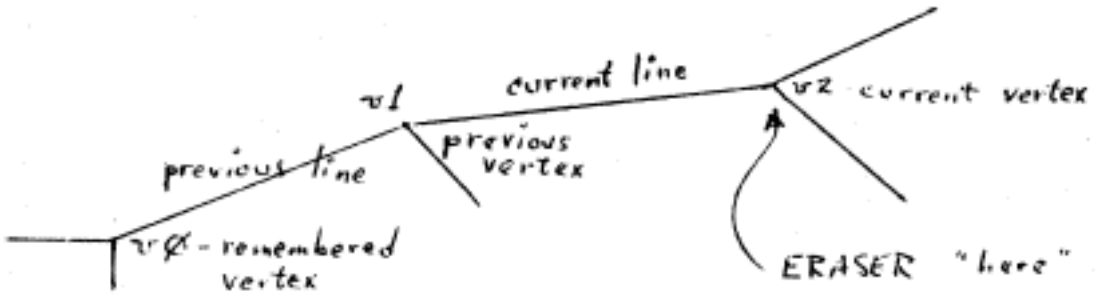



Figure 12.1a

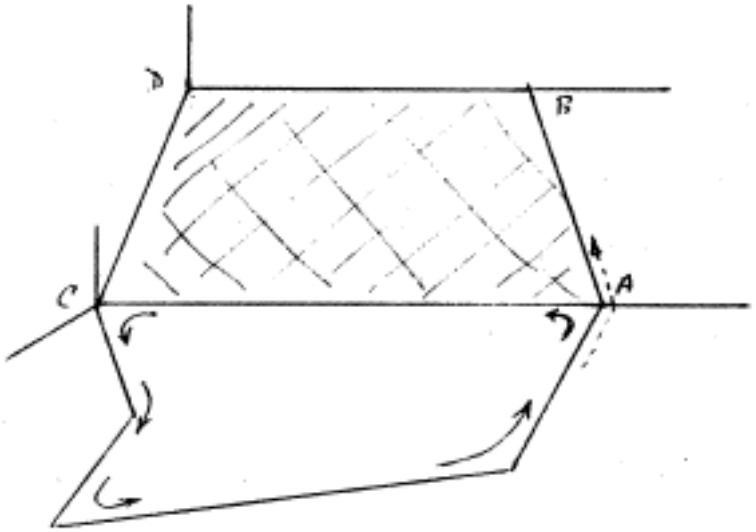



Figure 12.1b

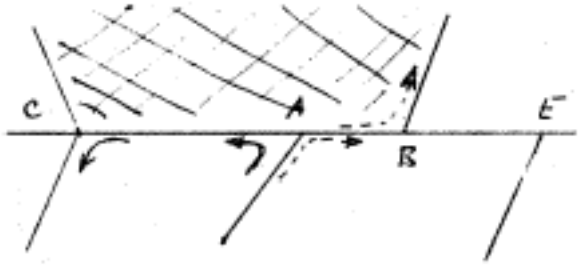



Figure 12.1c

**Figure 12.1.**

KEY

→ = normal direction of analysis

- - -→ = direction due to Series Two heuristic

▨ = added region

The notation for these diagrams is as follows:

the numbers correspond to regions

solid lines are strong links

dotted lines are weak links

the letters are the vertices causing the link.

The scenes considered in no way caused errors due to degenerate viewing conditions or optical illusions. The five cases show how the heuristics for solid objects are just unable to cope with shadows. The most devastating action caused by the presence of shadows is to divide a body into several pieces which would otherwise have been correctly recognized as continuous. This division is caused primarily by the heuristics connected with the T and X vertixes (overlapping bodies and interior edges) -- very basic heuristics which are quite powerful in noise-free situations.

Except for the difficulties created for later processing in the vision recognition sequence, the action of connecting a shadow with its body is technically correct. I think an indication of how three-dimensionally oriented SEE's heuristics are is the fact that shadows are merged together into nucleia as SEE tries to pull the two-dimensional shadows into a real body. Again, this is caused by the X and T heuristics.

The approach used in the design of ERASER was to categorize the characteristics of shadows into a set of heuristics that could

be applied to the vertex network prior to SEE. In this way the features caused by shadows, which run counter to the heuristics embodied in SEE, are completely removed from the network.

General Features of ERASER

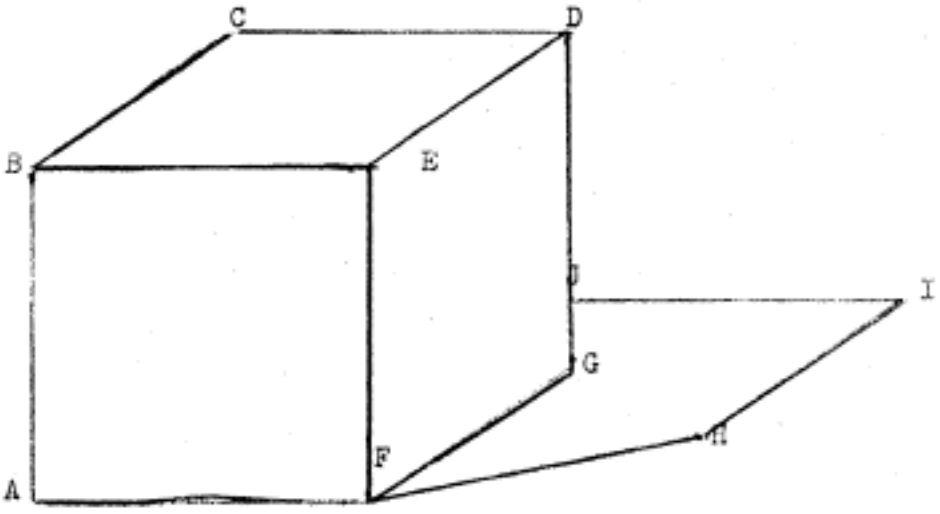
ERASER is to be used just prior to SEE, after the pre-processor has examined the simple vertex network and established the necessary vertex information. ERASER is handed a list of all the dark regions, the thought being that any shadows present in the scene will be contained in this list. Such information about the regions should be readily available from the light gradients given by the vidisector. Not all the dark regions are shadows, however, since the face of objects turned away from the light are as dark as shadows. And, some of the line segments bounding a shadow region also bound a real body region. Nevertheless, the heuristics in ERASER do not remove the real body lines, in most cases. Therefore, it would be reasonable to use ERASER as a predicate to decide if a given dark region were a shadow or not.

ERASER does not need to know global information about the scene, for example which body is causing the shadow, or light gradients across boundaries of the regions. The heuristics are entirely local and involve two, or at most three, connected vertices. Two types of heuristics are found in ERASER, grouped as Series One and

Series Two. Series One is further divided into primary and secondary.

The biases toward data that are built into ERASER are similar to those in SEE. With currently no mechanism set up to handle curved lines, ERASER assumes the connections in the vertex network are straight lines. Also, several of the predicate tests are designed with a right shadow projection in mind. However, a set for left-projected shadows could be made with a choice of which to use based on additional information supplied to ERASER. The projection-dependent tests are only used to effect a smoother operation and are not used in the decision to remove a suspected shadow.

The operational procedure is straightforward. ERASER examines the dark regions in the scene, one by one. Tracing counterclockwise around the perimeter of each such region, the sequence of vertex types encountered is analyzed. The primary heuristics of Series One determine if a line segment of the perimeter, connecting two vertices, is a shadow line or a part of a body. If believed to be a shadow, the segment is marked for later removal from the network. When all the vertices of a region have been considered in this manner a new region is analyzed. After the last region has been examined a function is called which removes the marked segments from the vertex network. As shown in Figure 13, vertices change type (J goes from an "I" to a straight "L", F goes from "MULTI" to "ARROW") or disappear (H, I -- shadow vertices of type "L").



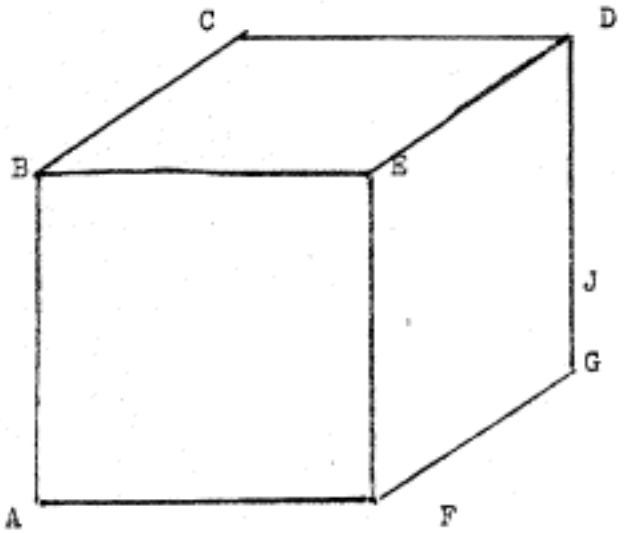




Figure 13.

The vertex network can now be collected and submitted to the preprocessor for reclassification of the modified vertices, before calling SEE. For Figure 13:

Before ERASER. . . . . (F (X Y) (H G E A))
(G (X Y) (F J))
(H (X Y) (F I))
(I (X Y) (H J))
(J (X Y) (G I D)) . . . . .

After ERASER. . . . . (F (x y) (G E A))
(G (x y) (F J))
(H (x y) NIL)
(I (x y) NIL)
(J (x y) (G D)) . . . . .

The secondary heuristics of Series One give an immediate evaluation of the primary heuristics. After the primary heuristics are finished and before the next vertex of the perimeter is brought into consideration, ERASER is looking at two line segments (see Figure 12.1a). At this point the secondary heuristics are called in to reconsider the decision made regarding the previous line segment. This decision is either accepted or reversed, based upon the current line segment (two vertices) just examined and the decision regarding it. This set of rules is most useful for removing the leading edge of a shadow (segment (F H) in Figure 13) or for

replacing part of a T that has been too quickly removed (segment (AN AK) in Figure 11). Modification of the vertex network tends to be an irreversible procedure and it is due to the secondary heuristics that the segments are only marked, and not removed, until all the regions of a scene have been examined.

To effect the operation of the Series One heuristics, a set of pattern-matching predicates were derived, based on two facts about shadows that allow their quick and easy recognition:

1. the abundance of Ls, Ts, and Xs bounding shadow regions,

2. the pattern of the occurrence of these three types of vertices in conjunction with the occurrence of the other possible types of vertices bounding any particular region.

These two facts are relevant in the following ways. First, the three vertex types mentioned are those caused most often by shadows. The L is the projection of a corner of a block located above the plane of the L. Normal intersections of shadows with bodies appear as Ts or Xs, e.g. when a body blocks the view, or a shadow or one body falls on two faces of another body. Second, the typical shadow has the following sequence of vertices as part of its perimeter (see Figure 13): ARROW - L - L - T. Of course, the viewing angle can, and does, affect the actual vertices of the shadow, with the exception of the Ls and Xs (other than obscuring them). Regular deviations from the typical case are to: replace

the T with an X or a K; insert a series of Ts and/or Xs between the Ls; or replace the ARROW with practically any type of vertex. When considering two vertices, one of which is of the type T, K, or X, it is necessary to know if the other vertex is in the continuing line of this T, K, or X. For example, in Figure 2, vertices A and C are in the continuing line of the "T" vertex B; vertices D and E are in the continuing line of "X" vertex C. A line segment that forms part of this continuing line cannot be a shadow line and is therefore never removed. Or, if removed by Series One/primary, it will be replaced by Series One/secondary. This rule is the basis for determining shadows and is specifically what the predicate tests for.

The Series Two heuristics are an attempt to make the analysis of the dark regions more realistic, i.e. more like what I imagine human analysis to be. Taking a hint from the way the X heuristic of SEE tends to pull shadows together, ERASER, through Series Two, tries to piece together dark regions to form a continuous shadow. As a result of these two types of heuristics the operation of ERASER is quite realistic.

An example: ERASER follows the shadow of the upright body in Figure 9 from the supporting surface, up the face of the solid object, across the top of the object, and then looks for the continuation of the shadow on the supporting surface, behind the long body. Being able to piece together the dark regions in this way

seems to yield a potentially valuable piece of information about the scene itself. One additional device could be added to improve this general semantic processing. ERASER should recognize that vertex AII presents an unnatural change of direction and that such changes of direction can be attributed to intersecting shadows. It would then jump up to vertex AV to continue its analysis, returning later to AII as the starting point for the second shadow. To be able to do this efficiently would require either more global heuristics or some type of pre-ERASER processing of the scene.

## Heuristics of ERASER

The primary and secondary heuristics embodied in the Series One group can be summarized as follows: (notation -- CV = "nm", the current vertex is type "nm"; PV = "nm", the previous vertex is type "nm")

### Primary

1. CV = L
   a. PV = FORK, ARROW, PEAK, K or MULTI; no action
   b. PV = L; remove the line
   c. PV = T and CV is not in the continuing line of the T; remove the line
2. CV = T
   a. PV = X or T and neither vertex is in the continuing line of

of the other; remove the line

b. PV = L and PV is not in the continuing line of CV; remove the line

c. PV = FORK, ARROW, PEAK, K or MULTI; no action

3. CV = X

a. PV = any and PV is not in the continuing line of CV; remove the line

4. CV = K is treated as a special case of CV = T above

5. CV = FORK, ARROW, PEAK or MULTI; no action

Secondary

6. If PV = T and the previous line was removed and the current vertex is in the continuing line of the PV; replace the previous line

7. If PV = L and the previous line was not removed and the current line has been removed; remove the previous line

8. If PV = X and CV is not in the continuing line of the PV and the current line has not already been removed; remove the current line.

The two heuristics of Series Two can be stated as follows:

1. When ERASER discovers the current vertex to be an X (Figure 12.1b, vertex A), rather than turn left as it does for, say, a MULTI (Figure 12.1b, vertex C) it continues on across the continuing line

of the X and adds the perimeter of the region left of segment AB to the perimeter of the region currently being analyzed.

2. A T vertex (Figure 12.1c, vertex A) causes ERASER to look right along the continuing line of vertex A for a T whose upright is oriented opposite to that of the current vertex (as shown by vertex B). Finding B, the region above AB and having segment AB as part of its perimeter is added to the current region. However, had E been the first vertex to the right, ERASER would have proceeded normally to vertex C.

As ERASER operates it documents which heuristic it invokes for a decision. The best way to demonstrate and explain these Series One and Two heuristics, as listed above, is to examine the printout from ERASER for some scenes.

The documentation ERASER yields is in three parts.

1. After printing its own name all the dark regions handed to ERASER are listed with:
   a. a name, viz. SHADOWn, where n is a number;
   b. a list of the vertices bounding the region, given in counterclockwise order.
2. When each dark region is considered the work HEURISTIC (the main function) is printed followed by the name of the region being considered. Immediately after this will be three types of information:

a. (v1 T1 v2 T2) -- v1 and v2 are the names of the vertices being analyzed; T1 and T2 are the types of the previous vertex, v1, and the current vertex, v2, respectively.

b. (T1 ---> T2) -- T1 and T2 are the same vertex types as in (a); however, this basic form will be in one of four formats. It shows whether a Primary or Secondary heuristic was applied (refer to Figure 12.1a).

i. T1====>T2 ; delete current line, primary (1)-(5)

ii. <******T1 ; restore previous line, secondary (6)

iii. ******>T1 ; delete previous line, secondary (7)

iv. T1******>T2 ; delete current line, secondary (8)

c. ((CALL RGN1 t) v1 & v2) This is printed when a Series Two heuristic is used. If t is X, then (i) was used; if t is T, (ii) was used; v1 and v2 are two vertices in the perimeter of the region being added to the current shadow. (RGN1 is the name of the function that actually does the merging.)

3. The final part is a listing of the results and is selfexplanatory.

Figure 13 shows the before and after plot of a simple scene. The ERASER printout is presented in Figure 14. For SHADOW1, the real body region, the no-action heuristics are (from the above charts): 1(a); 2(b) (since the L is in the continuing line of the T); 5; 5; 5. SHADOW2 is actually a shadow and three line segments are removed by these heuristics: 1(a); 1(b); 7; 2(b); 1(c); 5. Examples of the

```
ERASER

(SHADOW1 IS (F G J T F))
(SHADOW2 IS (F H I T G))

HEURISTIC (SHADOW1)
(F MULTI G L)
(G L J T)
(J T D ARROW)
(D ARROW E FORK)
(E FORK F MULTI)

HEURISTIC (SHADOW2)
(F MULTI H L)
(H L I L)
(L ======> L)
(*******> L)
(I L J T)
(L ======> T)
(J T G L)
(G L F MULTI)

RESULTS
(SHADOW1 : DELETED ; INF SEGMENTS ARE NIL)
(SHADOW2 : DELETED ; INF SEGMENTS ARE ((I J) (F H) (H I)))
NIL
```

Figure 14.

Series Two heuristics can be found in the printouts for both Figures 9 and 11. Figures 15 through 24 give the before/after plots of the five scenes shown earlier along with their diagnostic. The plots show the success of ERASER; SEE would now have no problems.

## All That Glitters...

The debugging of ERASER led to a couple of thoughts about when it will lose. Very dark blocks are definitely bad news. A light block and a dark block side by side look locally like a light block with a shadow on it. For such a situation shown in Figure 25a, if ERASER stumbled onto the boundary between the blocks (segments AB and BC), it would be removed. The supported block of Figure 21 shows the shadow consideration of this problem. I presently see no method for locally deciding this problem.

When a couple or three vertices are marked by the primary heuristics for removal and the continuing line rule remains satisfied, the line segments are quite dead. Various combinations of obscuring bodies will have the lethal sequence of vertex types if there are not sufficient shadows to break up the perimeters of the regions. In Figure 5, segment TU shows where a T-T sequence should be eliminated and segments AB and CD in Figure 25b show where the same sequence should not be removed. This again seems like a problem requiring a global solution.

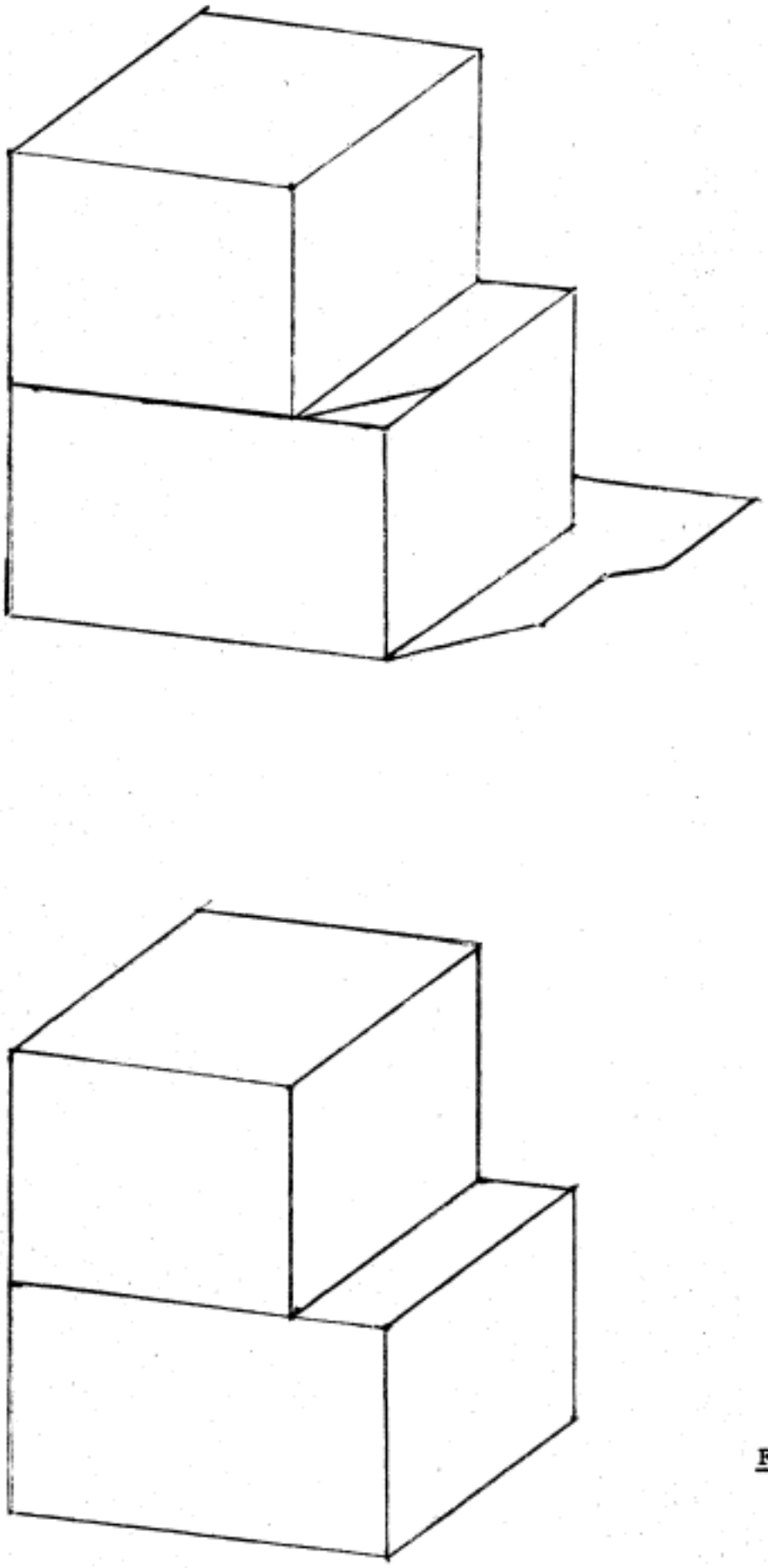
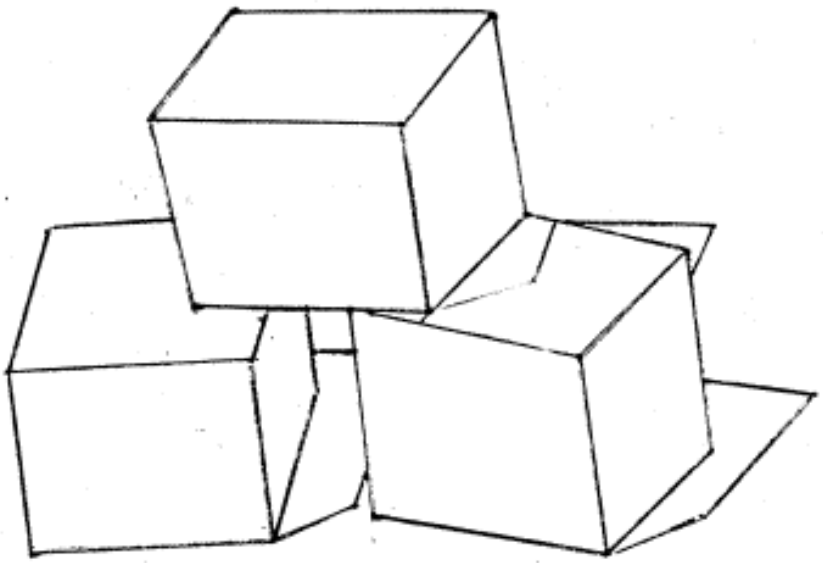


**Figure 15.**

```
ERASER

(SHADOW1 IS (G H E F))
(SHADOW2 IS (G J I H))
(SHADOW3 IS (L M N T J K))
(SHADOW4 IS (L R Q P O N M))

HEURISTIC (SHADOW1)
(G MULTI H FORK)
(H FORK E ARROW)
(E ARROW F FORK)
(F FORK G MULTI)

HEURISTIC (SHADOW2)
(G MULTI J T)
(MULTI ======> T)
(J T I ARROW)
(I ARROW H FORK)
(H FORK G MULTI)

HEURISTIC (SHADOW3)
(L MULTI M L)
(M L N T)
(N T I ARROW)
(I ARROW J T)
(J T K FORK)
(K FORK L MULTI)

HEURISTIC (SHADOW4)
(L MULTI R L)
(R L Q L)
(L ======> L)
(*******> L)
(Q L P L)
(L ======> L)
(P L O L)
(L ======> L)
(O L N T)
(L ======> T)
(N T M L)
(M L L MULTI)

RESULTS
(SHADOW1 : DELETED ; INF SEGMENTS ARE NIL)
(SHADOW2 : DELETED ; INF SEGMENTS ARE ((G J)))
(SHADOW3 : DELETED ; INF SEGMENTS ARE NIL)
(SHADOW4 : DELETED ; INF SEGMENTS ARE ((O N) (P O) (Q P) (L R) (R Q)))
NIL
```

Figure 16.



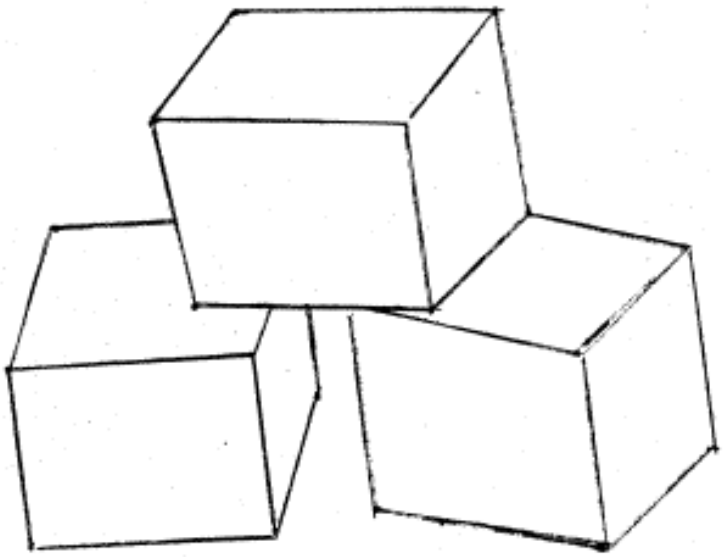
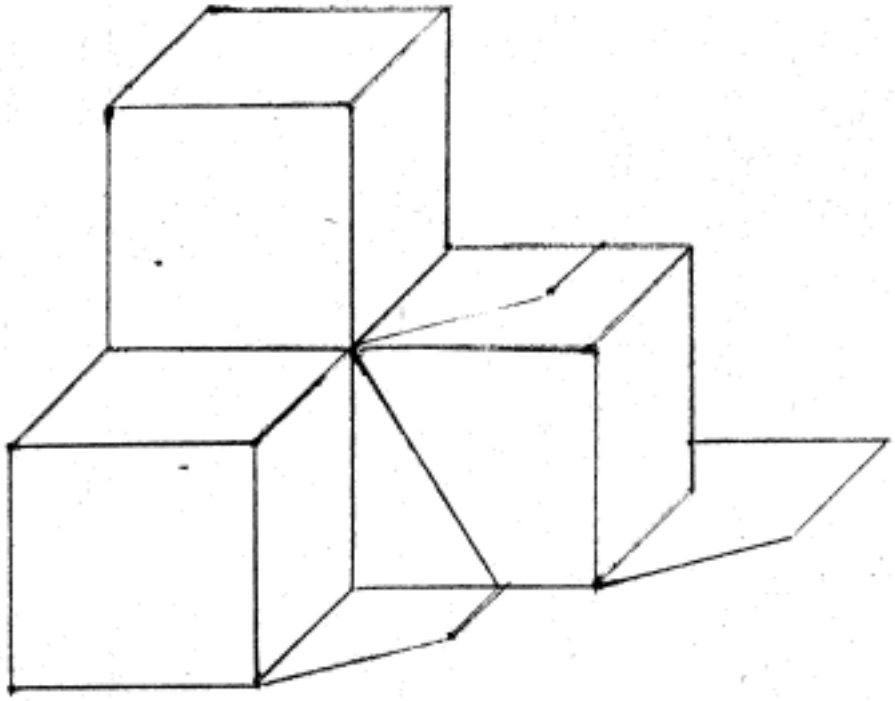


Figure 17.

```
FRASER

(SHADOW1 IS (E F T G H D))
(SHADOW2 IS (E S Q U T F))
(SHADOW3 IS (O V M N))
(SHADOW4 IS (O W X V))
(SHADOW5 IS (FF GG Y FF))
(SHADOW6 IS (Z CC DD FF EE Y))
(SHADOW7 IS (Z AA BB DD CC))

HEURISTIC (SHADOW1)
(E MULTI F L)
(F L T T)
(T T G T)
(G T H T)
(H T D FORK)
(D FORK E MULTI)

HEURISTIC (SHADOW2)
(E MULTI S L)
(S L Q T)
(L ======> T)
(******> L)
(Q T U T)
(<******* T)
(U T T T)
(T ======> T)
(T T F L)
(F L E MULTI)

HEURISTIC (SHADOW3)
(O MULTI V FORK)
(V FORK M ARROW)
(M ARROW N FORK)
(N FORK O MULTI)

HEURISTIC (SHADOW4)
(O MULTI W L)
(W L X X)
(L ======> X)
(******> L)
(X X V FORK)
(V FORK O MULTI)

HEURISTIC (SHADOW5)
(FF T GG L)
(T ======> L)
(GG L X X)
(L ======> X)
(X X EE ARROW)
(EE ARROW FF T)

HEURISTIC (SHADOW6)
(Z MULTI CC L)
(CC L DD T)
(DD T FF T)
(FF T EE ARROW)
(EE ARROW Y FORK)
(Y FORK Z MULTI)
```

```
HEURISTIC (SHADOW7)
(Z MULTI AA L)
(AA L BB L)
(L =====> L)
(******> L)
(BB L DD T)
(L =====> T)
(DD T CC L)
(CC L Z MULTI)

RESULTS
(SHADOW1 : DELETED ; INF SEGMENTS ARE NIL)
(SHADOW2 : DELETED ; INF SEGMENTS ARE ((U T) (F S) (S Q)))
(SHADOW3 : DELETED ; INF SEGMENTS ARE NIL)
(SHADOW4 : DELETED ; INF SEGMENTS ARE ((O W) (P Y)))
(SHADOW5 : DELETED ; INF SEGMENTS ARE ((GG X) (FF GG)))
(SHADOW6 : DELETED ; INF SEGMENTS ARE NIL)
(SHADOW7 : DELETED ; INF SEGMENTS ARE ((BB DD) (Z AA) (AA BB)))
NIL
```

Figure 18.



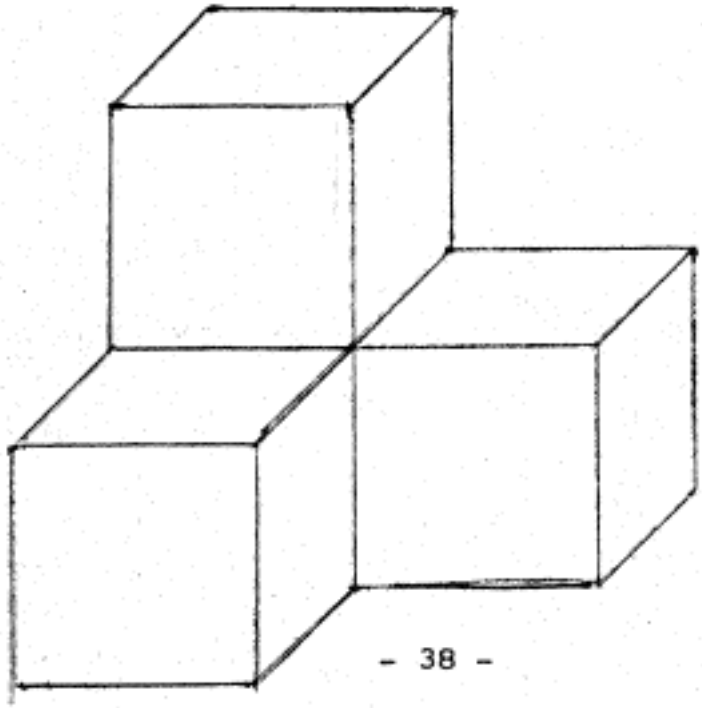
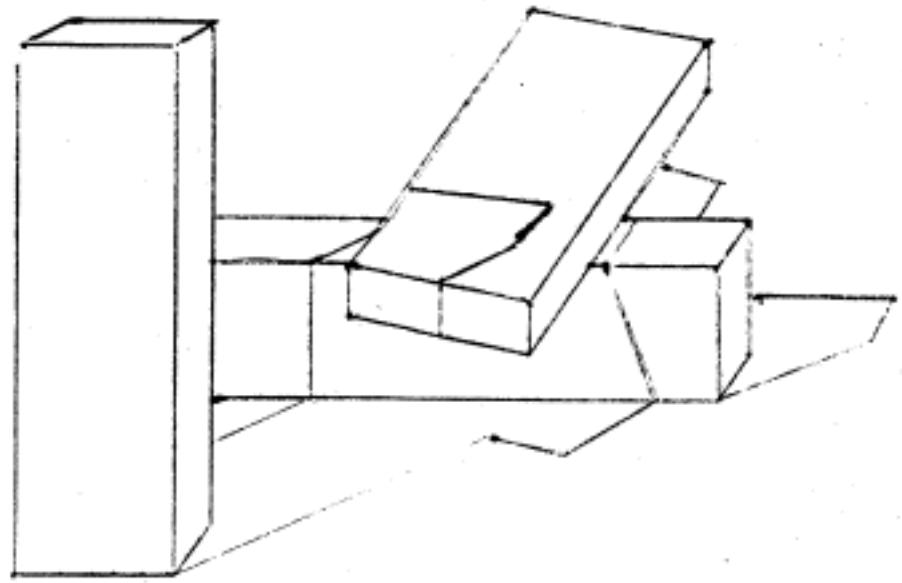


Figure 19.

```
FRASER

(SHADOW1 IS (E G F D))
(SHADOW2 IS (E P O G))
(SHADOW3 IS (G O F))
(SHADOW4 IS (F L J K))
(SHADOW5 IS (F N M L))
(SHADOW6 IS (O U T S R))
(SHADOW7 IS (O W V T U))

HEURISTIC (SHADOW1)
(F MULTI G FORK)
(G FORK F MULTI)
(F MULTI D FORK)
(D FORK F MULTI)

HEURISTIC (SHADOW2)
(E MULTI P L)
(P L O X)
((CALL RGN1 X) O R F)
(L ======> X)
(*******> L)
(O X F MULTI)
(X *******> MULTI)
(F MULTI G FORK)
(G FORK F MULTI)

HEURISTIC (SHADOW3)

HEURISTIC (SHADOW4)
(F MULTI L FORK)
(L FORK J ARROW)
(J ARROW K FORK)
(K FORK F MULTI)

HEURISTIC (SHADOW5)
(F MULTI N L)
(N L M T)
(L ======> T)
(*******> L)
(M T L FORK)
(L FORK F MULTI)

HEURISTIC (SHADOW6)
(O MULTI U L)
(U L T T)
(T T S ARROW)
(S ARROW R FORK)
(R FORK O MULTI)

HEURISTIC (SHADOW7)
(O MULTI W L)
(W L V L)
(L ======> L)
(*******> L)
(V L T T)
(L ======> T)
(T T U L)
(U L O MULTI)
```

```
RESULTS
(SHADOW1 : DELETED (INF SEGMENTS ARE NIL)
(SHADOW2 : DELETED (INF SEGMENTS ARE ((O F) (F P) (P O)))
(SHADOW3 : ABSORBED INTO SHADOW2)
(SHADOW4 : DELETED (INF SEGMENTS ARE NIL)
(SHADOW5 : DELETED (INF SEGMENTS ARE ((F N) (N M)))
(SHADOW6 : DELETED (INF SEGMENTS ARE NIL)
(SHADOW7 : DELETED (INF SEGMENTS ARE ((V T) (C W) (W V)))
NIL
```

Figure 20.



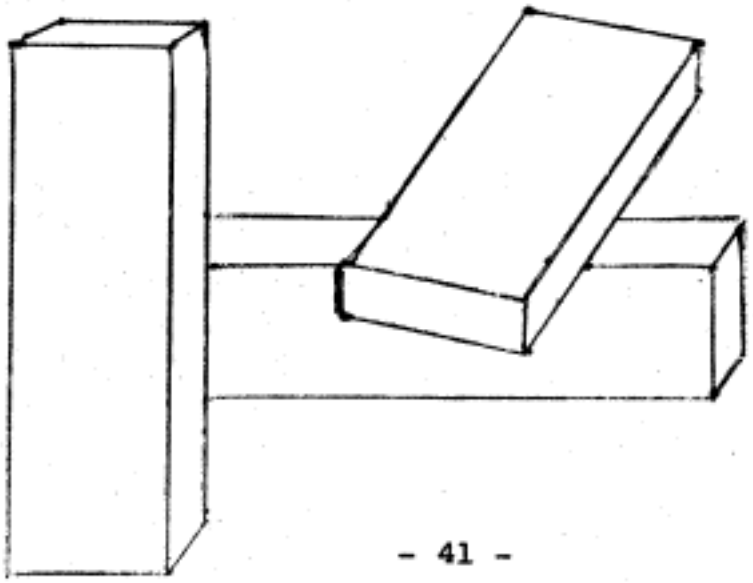


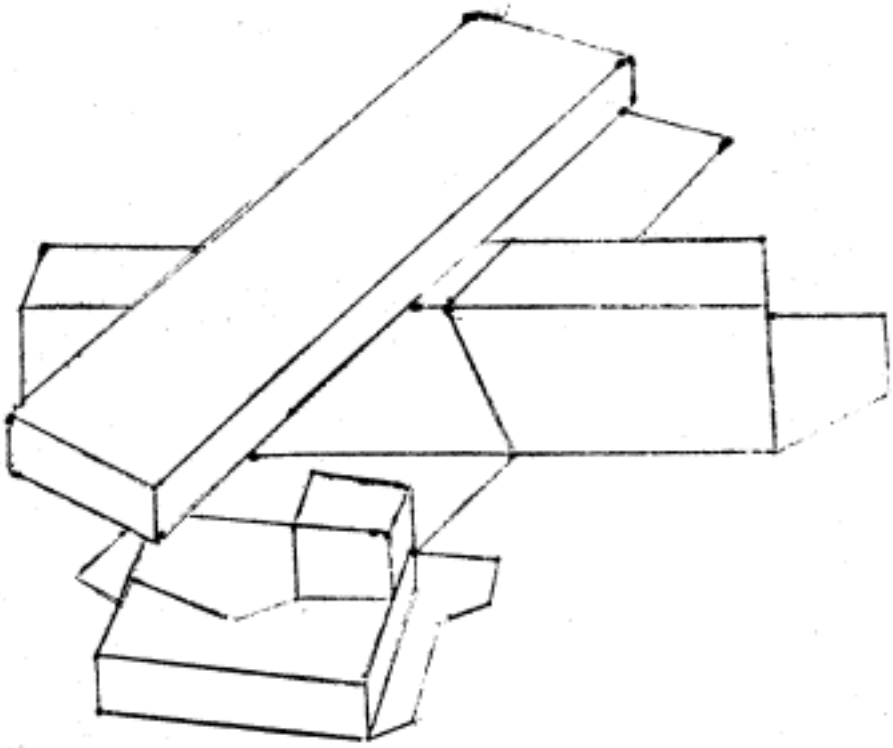
Figure 21.

```
ERASER

(SHADOW1 IS (AF AG AH AI AJ AK AC AD))
(SHADOW2 IS (AF AII AJJ AKK AQ AH AG))
(SHADOW3 IS (AQ AKK AFF AGG AW AV AR AP AD))
(SHADOW4 IS (AR AV AU AP))
(SHADOW5 IS (AP AN AO))
(SHADOW6 IS (AP AU AT AS AL AM AN))
(SHADOW7 IS (AW AGG AHH ARR AAA AZ AY))
(SHADOW8 IS (AGG AEE AFF AHH))
(SHADOW9 IS (AEE ADD ACC AHR AHH))
(SHADOW10 IS (ARR AQQ ANN AMM ALI))
(SHADOW11 IS (ARR APP AQQ ANN AQ4))

HEURISTIC (SHADOW1)
(AF PEAK AG L)
(AG L AH T)
(AH T AI T)
(AI T AJ T)
(AJ T AK T)
(AK T AC ARROW)
(AC ARROW AD FORK)
(AD FORK AF PEAK)

HEURISTIC (SHADOW2)
(AF PEAK AII L)
(AII L AJJ L)
(L ======> L)
(*******> L)
(AJJ L AKK X)
((CALL RGN1 X) AKK ? AFF)
(L ======> X)
(AKK X AFF X)
((CALL RGN1 X) AFF ? AEE)
(X ======> X)
(AFF X AEE T)
((CALL RGN1 T) AEE ? ADD)
(X ======> T)
(AEE T ADD T)
(ADD T ACC L)
(T ======> L)
(ACC L ARR T)
(L ======> T)
(ARR T AHH T)
(<******* T)
(AHH T AGG T)
(AGG T AW ARROW)
(AW ARROW AV T)
(AV T AR L)
(AR L AP MULTI)
(AP MULTI AO X)
(AO X AQ X)
(X ======> X)
(AQ X AH T)
(X ======> T)
(AH T AG L)
(AG L AF PEAK)

HEURISTIC (SHADOW3)
```

```
HEURISTIC (SHADOW4)
(AR L AV T)
(AV T AU X)
((CALL RGN1 X) AU & AT)
(T ======> X)
(AU X AT L)
(X ======> L)
(AT L AS L)
(L ======> L)
(AS L AL T)
(L ======> T)
(AL T AM T)
(AM T AN T)
(AN T AP MULTI)
(AP MULTI AR L)

HEURISTIC (SHADOW5)
(AP MULTI AN T)
(AN T AO X)
(T ======> X)
(AO X AP MULTI)

HEURISTIC (SHADOW6)

HEURISTIC (SHADOW7)
(AW ARROW AGG T)
(AGG T AHH T)
(AHH T ABB T)
(ABB T AAA L)
(AAA L AZ ARROW)
(AZ ARROW AX FORK)
(AX FORK AW ARROW)

HEURISTIC (SHADOW8)

HEURISTIC (SHADOW9)

HEURISTIC (SHADOW10)
(ARR PEAK AQQ L)
(AQQ L ANN T)
(ANN T AMM ARROW)
(AMM ARROW ALL FORK)
(ALL FORK ARR PEAK)

HEURISTIC (SHADOW11)
(ARR PEAK APP L)
(APP L AQQ L)
(L ======> L)
(*******> L)
(AQQ L ANN T)
(L ======> T)
(ANN T AQQ L)
(AQQ L ARR PEAK)

RESULTS
(SHADOW1 : DELETED ; INF SEGMENTS ARE NIL)
(SHADOW2 : DELETED ; INF SEGMENTS ARE ((AO AH) (AO AG) (ADD ACC) (ACC ARR
) (AFF AEE) (AKK AFF) (AJJ AKK) (AF AII) (AII AJJ)))
(SHADOW3 : ABSORBED INTO SHADOW2)
```

```
(SHADOW4 : DELETED ; INF SEGMENTS ARE ((AS AL) (AU AT) (AT AS) (AV AU)))
(SHADOW5 : DELETED ; INF SEGMENTS ARE ((AN AO)))
(SHADOW6 : ABSORBED INTO SHADOW4)
(SHADOW7 : DELETED ; INF SEGMENTS ARE NIL)
(SHADOW8 : ABSORBED INTO SHADOW2)
(SHADOW9 : ABSORBED INTO SHADOW2)
(SHADOW10 : DELETED ; INF SEGMENTS ARE NIL)
(SHADOW11 : DELETED ; INF SEGMENTS ARE ((AOO AMN) (ARR APP) (APP AOO)))
NIL
```

Figure 22.



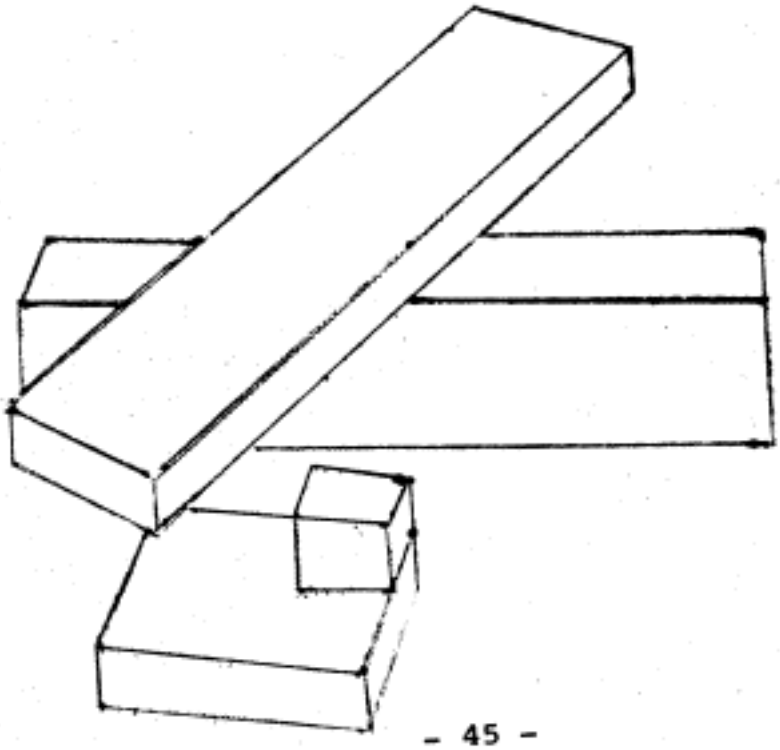


Figure 23.

```
ERASER

(SHADOW1 IS (AL AM AN AK AH))
(SHADOW2 IS (AN AR AS AT AQ AJ AK))
(SHADOW3 IS (ABB ACC ADD AFF AGG AZ AEE))
(SHADOW4 IS (ABB AEE AZ AY AX AAA))
(SHADOW5 IS (AZ AII AHH AQ AI AU AW AY))
(SHADOW6 IS (AHH AII AKK AJII))
(SHADOW7 IS (AYY AXX ANN AVV))
(SHADOW8 IS (AX AY AN AV))
(SHADOW9 IS (AHH AJI AMM APP AQQ ARR AI AJ AQ))
(SHADOW10 IS (AJI AKK ALL AMM))
(SHADOW11 IS (ALI ANN AQQ APP AMM))

HEURISTIC (SHADOW1)
(AL L AM T)
(L ======> T)
(*******> L)
(AM T AN T)
(AN T AK T)
(AK T AH T)
(AH T AL L)
(T ======> L)

HEURISTIC (SHADOW2)
(AN T AR L)
(T ======> L)
(AR L AS FORK)
(AS FORK AT MULTI)
(AT MULTI AQ T)
(MULTI ======> T)
(AQ T AJ ARROW)
(AJ ARROW AK T)
(AK T AN T)

HEURISTIC (SHADOW3)
(ABB PEAK ACC L)
(ACC L ADD L)
(L ======> L)
(*******> L)
(ADD L AFF L)
(L ======> L)
(AFF L AGG L)
(L ======> L)
(AGG L AZ K)
(L ======> K)
(AZ K AEE L)
(AEE L ABB PEAK)

HEURISTIC (SHADOW4)
(ABB PEAK AEE L)
(AEE L AZ K)
(AZ K AY T)
(AY T AX K)
(AX K AAA FORK)
(AAA FORK ABB PEAK)

HEURISTIC (SHADOW5)
(AZ K AII X)
```

```
((CALL RGN1 X) AII * AKK)
(K ======> X)
(AII X AKK X)
((CALL RGN1 X) AKK * ALL)
(X ======> X)
(AKK X ALL T)
((CALL RGN1 T) ALL * ANN)
(X ======> T)
(ALL T ANN T)
(ANN T AOO L)
(T ======> L)
(AOO L APP T)
(L ======> T)
(APP T AMM T)
(<******* T)
(AMM T AJJ T)
(AJJ T AHH T)
(AHH T AQ T)
(AQ T AT MULTI)
(AT MULTI AU L)
(AU L AW ARROW)
(AW ARROW AY T)
(AY T AZ K)

HEURISTIC (SHADOW6)

HEURISTIC (SHADOW7)
(AYY ARROW AXX L)
(AXX L AXW L)
(L ======> L)
(*******> L)
(AWW L AVV T)
(L ======> T)
(AVV T AYY ARROW)

HEURISTIC (SHADOW8)
(AX K AY T)
(AY T AW ARROW)
(AW ARROW AV FORK)
(AV FORK AX K)

HEURISTIC (SHADOW9)
(AHH T AJJ T)
(AJJ T AMM T)
(AMM T APP T)
(APP T AQQ L)
(AQQ L ARR ARROW)
(ARR ARROW AI FORK)
(AI FORK AJ ARROW)
(AJ ARROW AQ T)
(AQ T AHH T)
(<******* T)

HEURISTIC (SHADOW10)

HEURISTIC (SHADOW11)

RESULTS
(SHADOW1 : DELETED LINE SEGMENTS ARE ((AW AI) (AL AM)))
(SHADOW2 : DELETED LINE SEGMENTS ARE ((AN AP) (AR AS)))
```

```
(SHADOW3 : DELETED LINE SEGMENTS ARE ((AGG AZ) (AFF AGG) (ADD AFF) (ABB
ACC) (ACC ADD)))
(SHADOW4 : DELETED LINE SEGMENTS ARE NIL)
(SHADOW5 : DELETED LINE SEGMENTS ARE ((ANN ADD) (ADD APP) (AKK ALL) (ATT
 AKK) (AZ ATT)))
(SHADOW6 : ABSORBED INTO SHADOW5)
(SHADOW7 : DELETED LINE SEGMENTS ARE ((AWW AVV) (AYY AXX) (AXX AWW)))
(SHADOW8 : DELETED LINE SEGMENTS ARE NIL)
(SHADOW9 : DELETED LINE SEGMENTS ARE NIL)
(SHADOW10 : ABSORBED INTO SHADOW5)
(SHADOW11 : ABSORBED INTO SHADOW5)
NIL
```

Figure 24.

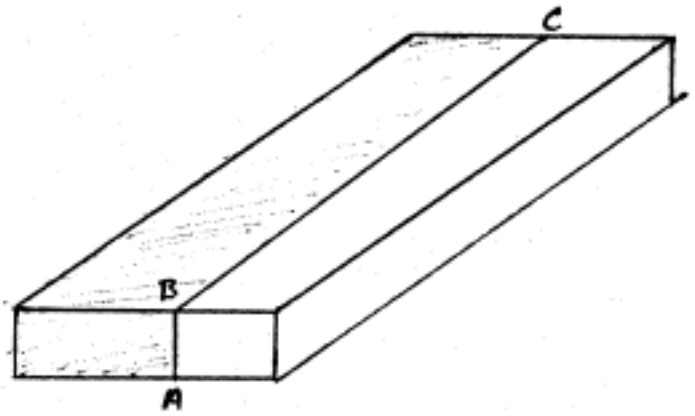



Figure 25a

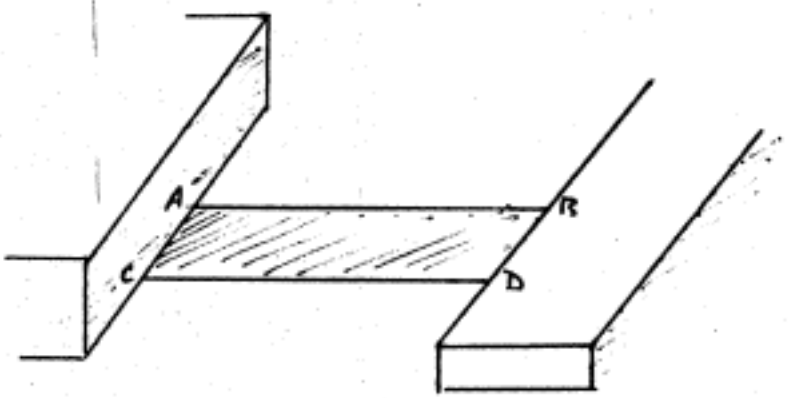



Figure 25b

```
(NOTE:  "FOO" TAKES THE PLACE OF UNUSED DATA
        "NVERTICES" IS THE NEIGHBORS PROPERTY)

VERTICES

(A B C D E F G H I J K L M N O P Q R)


A
(TYPE FOO TYPE* L NVERTICES (L B) YCOR 30 XCOR 10 PNAME (#40650))

B
(TYPE (T (G B A C *)) TYPE* T NVERTICES (A G C) YCOR 100 XCOR 10 PNAME (
#40647))

C
(TYPE FOO TYPE* ARROW NVERTICES (B F D) YCOR 130 XCOR 10 PNAME (#40645))


D
(TYPE FOO TYPE* L NVERTICES (C E) YCOR 160 XCOR 50 PNAME (#40642))

E
(TYPE FOO TYPE* ARROW NVERTICES (F H D) YCOR 152 XCOR 110 PNAME (#40640)
)

F
(TYPE FOO TYPE* FORK NVERTICES (G E C) YCOR 122 XCOR 70 PNAME (#40643))

G
(TYPE FOO TYPE* MULTI NVERTICES (K J H F B) YCOR 72 XCOR 70 PNAME (#4065
6))

H
(TYPE FOO TYPE* FORK NVERTICES (G I E) YCOR 102 XCOR 110 PNAME (#40655))


I
(TYPE FOO TYPE* ARROW NVERTICES (N H J) YCOR 100 XCOR 130 PNAME (#40634)
)

J
(TYPE (T (G J I K **)) TYPE* T NVERTICES (I G K) YCOR 100 XCOR 103 PNAME
 (#40654))

K
(TYPE FOO TYPE* FORK NVERTICES (J G L) YCOR 70 XCOR 110 PNAME (#40636))

L
(TYPE FOO TYPE* MULTI NVERTICES (R M K A) YCOR 20 XCOR 110 VALUE (NIL .
L) PNAME (#40653))
```

```
M
(TYPE FOO TYPE* L NVERTICES (L N) YCOR 50 XCOR 130 PNAME (#40652))

N
(TYPE (T (O N M I *)) TYPE* T NVERTICES (N O I) YCOR 60 XCOR 130 PNAME (
#40633))

O
(TYPE FOO TYPE* L NVERTICES (P N) YCOR 55 XCOR 170 PNAME (#40631))

P
(TYPE FOO TYPE* L NVERTICES (O Q) YCOR 40 XCOR 151 PNAME (#40630))

Q
(TYPE FOO TYPE* L NVERTICES (P R ) YCOR 40 XCOR 137 PNAME (#40627))

R
(TYPE FOO TYPE* L NVERTICES (Q L) YCOR 30 XCOR 123 PNAME (#40651))
```

**Figure 26.**

How to Use It

The complete ERASER is contained on the file named "ERASER n" where n is the highest integer found. Figure 26 is a printout of all the information about the scene of Figure 3 that is needed for successful operation of ERASER. Each vertex is named and has been given an x and y coordinate, a TYPE, a TYPE*, and a NEIGHBORS list. These properties are explained in the paper by Guzman. The argument to ERASER is a list of line segments. A segment is a line from the perimeter of one dark region, and is represented by the list of its two end points. ERASER expands the line segment into the complete perimeter of the dark region and begins computation.

If a vertex is the endpoint of a segment that is to be removed, the other endpoint of the segment is put on its property list under the indicator KILL. At segment removal time this KILL vertex is removed from the NEIGHBORS list.

Acknowledgement

I gratefully acknowledge the suggestions, comments, criticisms, and encouragement (not to mention patience) of Thomas Binford, who started me on this project and often redirected my efforts toward the desired goal.

References

1. Guzman, A. Decomposition of a Visual Scene into Bodies, MAC-M-357, A.I. Memo 139, Project MAC, MIT (September 1967).

2. Mahabala, H.N. Pre-Processor for Programs Which Recognize Scenes, A.I. Memo 177, Project MAC, MIT (August 1969).

3. Winston, P. HOLES. A.I. Memo 163, Project MAC, MIT (August 1968).