COMPUTER RECOGNITION OF THREE-DIMENSIONAL OBJECTS IN A VISUAL SCENE

by

ADOLFO GUZMÁN - ARENAS

I. C. E. Instituto Politécnico Nacional
(ESIME) México, 1965

S. M. Massachusetts Institute of Technology
1967

SUBMITTED IN PARTIAL FULFILLMENT OF THE

REQUIREMENTS FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

December 1968, i.e. Feb., 1969

Signature redacted

Signature of Author . . . .
Department of Electrical Engineering, Dec. 1968

Signature redacted

Certified by . . . .
Thesis Supervisor

Signature redacted

Accepted by . . . .
Chairman, Departmental Committee on Graduate Student

COMPUTER RECOGNITION OF THREE-DIMENSIONAL OBJECTS

IN A VISUAL SCENE

by

A d o l f o   G u z m á n - A r e n a s

Submitted to the Department of Electrical Engineering on December 30, 1968 in partial fulfillment of the requirements for the Degree of Doctor of Philosophy

A B S T R A C T

Methods are presented (1) to partition or decompose a visual scene into the bodies forming it; (2) to position these bodies in three-dimensional space, by combining two scenes that make a stereoscopic pair; (3) to find the regions or zones of a visual scene that belong to its background; (4) to carry out the isolation of objects in (1) when the input has inaccuracies. Running computer programs implement the methods, and many examples illustrate their behavior. The input is a two-dimensional line-drawing of the scene, assumed to contain three-dimensional bodies possessing flat faces (polyhedra); some of them may be partially occluded. Suggestions are made for extending the work to curved objects. Some comparisons are made with human visual perception.

The main conclusion is that it is possible to separate a picture or scene into the constituent objects exclusively in basis of monocular geometric properties (in basis of pure form); in fact, successful methods are shown.

Thesis Supervisor: Marvin L. Minsky.
Title: Professor of Electrical Engineering.

ACKNOWLEDGEMENTS

I sincerely appreciate the constant guidance and encouragement of Professors Marvin L. Minsky (thesis supervisor) and Seymour A. Papert. The pertinent criticism of Professor Joseph C. R. Licklider is gratefully appreciated.

Thanks are extended to Miss Cornelia A. Sullivan and Mr. Devendra D. Mehta for their kind assistance to provide material used here and their help in the preparation of this thesis, and to the many friends that made this work possible. Special thanks to the Instituto Nacional de la Investigación Científica (México), who partially supported me.

Signature redacted

Project MAC, M. I. T.
545 Technology Square
Cambridge, Mass., USA., December 30, 1968.

T A B L E   O F   C O N T E N T S

For a quick glance at this thesis, follow directions in page 14

SECTION	PAGE
ABSTRACT	2
ACKNOWLEDGMENTS	3
TABLE OF CONTENTS	4
INTRODUCTION — The scope of the thesis is presented	10 - 13
Purpose	10
Why this work was chosen as a thesis topic	12
SIMPLIFIED VIEW OF SCENE ANALYSIS — A general view of the problems in the thesis and their solutions	14 - 35
Scene Analysis	14
Recognition	18
Analysis of several examples	22
Statement of Rules	25
Conclusion	30
Problems in analyzing a visual scene	31
Other projects	32
RELATED RESEARCH	33
Previous work by the author	33
Convert	33
Scene Analysis	33
Canaday	34
Roberts	35
Mechanical manipulator groups	35
THE CONCEPT OF A BODY — Definitions of a body or object will be proposed and discussed	36 - 57
Introduction	36
The problem is inherently ambiguous	36
Sibelius' monument	38
Legal scene	39
Metatheorem	39
Trivial partition	41
Simplicity criterion	42
Other kinds of two-dimensional data	43
Conclusion	43

TOTAL ANALYSIS OF VERTICES 44
Synopsis 44
Vertices are the important feature 44
Genuine and false vertices 46
Problems to be solved 47
Classification of vertices 48
"Theorem" 50
Generation of partitions 51
Digression 1. An alternate approach 57

SEE, A PROGRAM THAT FINDS BODIES IN A SCENE 58 - 103

It is explained how SEE works

Synopsis 58
INTRODUCTION 60
Division of work in computer vision 60
Technical descriptions of SEE 61
INPUT FORMAT 63
Property lists in Lisp 64
INTERNAL FORMAT 66
Region 66
Vertex 67
TYPES OF VERTICES 70
Vertices where two lines meet 70
Vertices where three lines meet 70
Vertices where four lines meet 72
Other types of vertices 73
Nextes or matching T's 73
THE PROGRAM 78
Example A. TOWER 78
Example B. MOMO 78
Example C. R3 79
The parts of SEE 80
Auxiliary routines (Throughtes, Goodt, Nosabo) 81
LINK FORMATION 83
NUCLEI CONSOLIDATION 91
BODY RETOUCHING 92
Example. HARD 100
RESULTS 103
Summary 103

ANALYSIS OF MANY SCENES 104 - 182

(A list of scenes analyzed by SEE is given in page 106)
Discussion 182

CURVED OBJECTS 183 - 190

How to extend SEE to work with objects possessing curved surfaces

Introduction and Summary 183
At some point, we have to know what we want 186
APPENDIX TO SECTION ON CURVE OBJECTS 187
Requirements for the preprocessor 187
How bad will be curved objects 187
Additional information could be used 187
Psychological evidence 187

ON OPTICAL ILLUSIONS 191 - 205

Performance of SEE on misleading images

Three kinds of illusions 191
POSSIBLE BUT NOT "GOOD" INTERPRETATION 191
AMBIGUOUS - TWO GOOD INTERPRETATIONS 196
IMPOSSIBLE: WITHOUT INTERPRETATION 198
A PROGRAM TO DISCOVER HUMAN OPTICAL ILLUSIONS 199
H·optical illusions 200
A program to discover h·optical illusions 200
How to solve equations (E) 203
Conclusions and conjectures 204

ON NOISY INPUT 206 - 221

Performance of SEE on inaccuracies

Summary 206
OBTAINING THE DATA 207
MISPLACED VERTICES 211
Equal within epsilon 215
Tolerances in collinearity and parallelism 215
Straightening twisted segments 216
If the information is very bad 216
Summary 217
MISSING EDGES 217
Illegal scenes 217
Line proposer and line verifier 217
Blum's line proposer 218
Internal edges 218
External edges 219
SPURIOUS EXTRA LINES 220
MERGED VERTICES 221
CONCLUSION 221

BACKGROUND DISCRIMINATION BY COMPUTER 222 - 232

A program determines the regions that belong to the background of a scene

Need 222
Suspicious 223
Clean vertex 224
Summary 226
More global indications 227
Other examples of background finding 230
The problem is ambiguous 232
Summary 232
Conclusion 232

STEREO PERCEPTION 233 - 247

The problem of locating the objects in three-dimensional space

Summary 233
Theorem S-2 234
When the optical axes are parallel 238
USE OF SEE IN STEREO PERCEPTION 238
Summary 243
Scene L10 - R10 246

CONCLUSIONS 248 - 255

Looking behind 248
Looking ahead 248
General notation 250
Use 250
Assigning a name to an object 251
Do not use over-specialized assumptions 252
Other example of over-specialization 253
Conclusion 253
Human perception versus computer perception 254
TABLE "ASSUMPTIONS" 255

List of Suggestion Boxes 256

References 257 - 259

Annotated listing of the functions used 260 - 284

Alphabetical index

Biographical Note 285 - 287

LIST OF FIGURES AND TABLES 8 - 9

LIST OF FIGURES

NAME	PAGE
ARCH	165
BACKGROUND	223
BLACK	194
BLUM	218
BRIDGE	28, 94, 180
CAUTION	47
CHURCH	37
CONTINUATION	48
CONTRADICTORY	199
CORN	230, 150
CROSSED	215
CUBE	39
DESCRIPTIVE	253
DISCONNECTED	212
EQUILIBRIUM	226
EXTERNAL EDGES	220
FINAL-BRIDGE	98
FRUIT	186
γ-PARAMETRIZATION	236
GENUINE	45
GENUINE PAIRS	235
HARD	101, 223, 168
HOLES	252
ILLEGAL	217
INTERVALS	241
LIGHT AND SHADOW	220
LINKS-BRIDGE	95
L2	141
L3	111
L4	171
L10	59, 124
L10 - R10	247
L12	244
L13	239
L19	144
L9	153
MACHINE	50
MISSING	219
MOMO	29, 77, 231, 177
NEW-NUCLEI-BRIDGE	97
NODES	46
NUCLEI-BRIDGE	96
PARALLELEPIPED	40
PENROSE TRIANGLE	192
POINTS	234
REWOT	133
R2	139
R3	62, 114
R4	174
R9	158
R9T	159
R10	127
R12	245
R13	240
R17	108
R19	147
SIBELIUS	38
SPREAD	117, 227
STACK	120
STACK*	121
STAIRCASE	196
SUITCASES	185
TEST OBJECTS	208
TEST OBJECTS (cont)	209
TOWER	76, 130
TRIAL	88, 89, 162
TRIAL-FINAL	92
TRIAL-LINKS	90
TRIAL-NUCLEI	92
TWISTED	216
VARIANT	194
VIOLATIONS	225
WRIST*	136

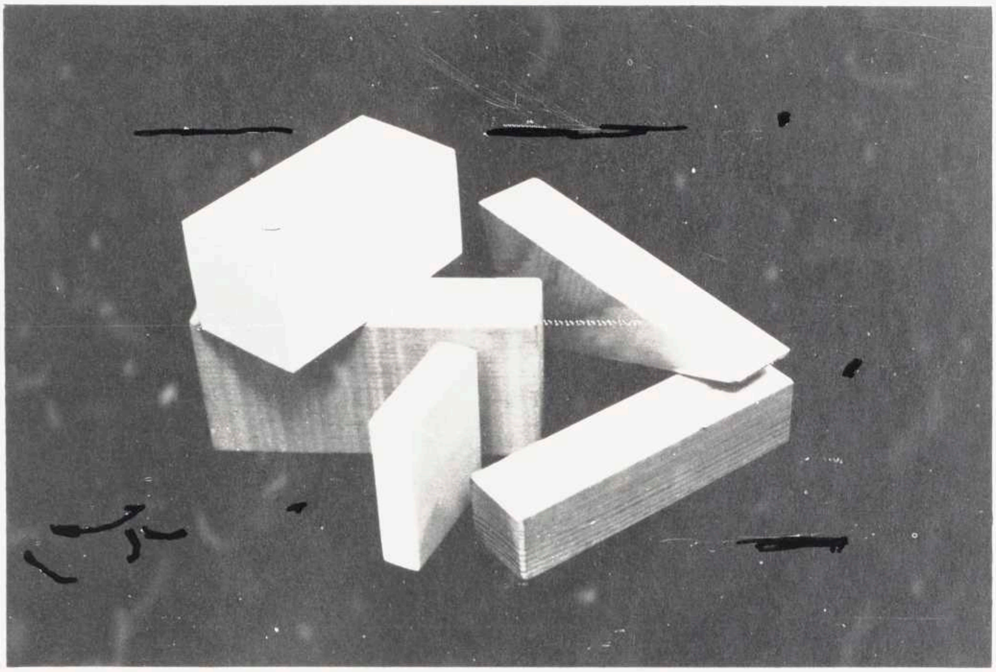
LIST OF TABLES

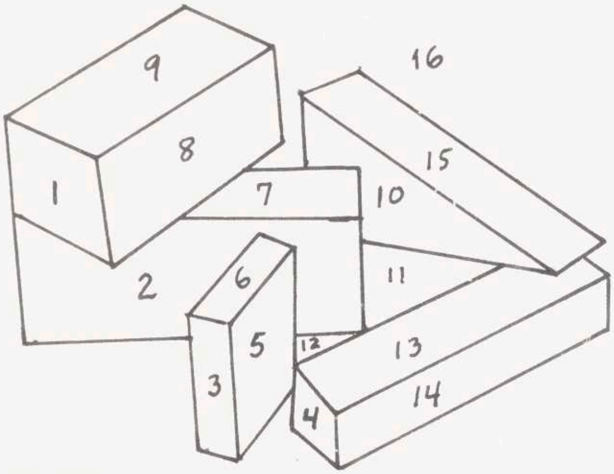
ASSUMPTIONS	255
BOX	201
DESIRABILITY CRITERION	212
GLOBAL EVIDENCE	87
R3 IN INTERNAL FORMAT	65
THREE T's	214
VERTICES	49, 69

Purpose

== This thesis explains how a computer can find, identify and recognize objects in a visual scene. For instance, when analyzing the following scene,

AUG · 68





if the machine is asked to separate the bodies, it must say

(BODIES ARE AS FOLLOWS : (1 8 9) (2 7) (3 5 6) (10 15) (4 13 14))

If asked to report the triangular prisms, it should answer

(10 15 IS A TRIANGULAR PRISM)

== This thesis discusses the problems involved in this task.

What should be done when the information is noisy, some lines are missing, etc?

How can the computer separate the background from the objects forming the scene?

How should shadows be handled?

How can stereoscopic vision be used?

What about ambiguities and optical illusions?

== This thesis touches some aspects of Psychology in regard to human visual perception

== Key words and phrases related to this study are as follows:

artificial intelligence
body
background
background discrimination
classification of images
CONVERT
cybernetics
feature recognition
geometric objects
geometric processing
graphic processing
graphical communication
graphical data
heuristic procedures
heuristic programming
identification
image
intelligence
line drawing
LISP
list processing
machine aided cognition
machine perception
mechanization of visual perception
object identification
optical
optical illusion
pattern
pattern matching
pattern recognition
photography
photo-interpretation
picture
picture abstraction
picture processing
picture transformations
pictorial structures
polyhedra
recognition
robot
scene
scene analysis
solids
stereoscopic
symbol manipulation
three-dimensional
three-dimensional scenes
three-dimensional solids
two-dimensional patterns
vision
visual
visual information processing
visual object recognition
visual perception
visual scenes

== Computer Review (A. C. M.) index numbers: C.R. 3.61, 3.63, 4.22, 5.20.

<u>Why this work was chosen as a thesis topic</u> The present work was carried out in the facilities of the Artificial Intelligence Group of Project MAC, at M. I. T. Currently, the main goal of the Artificial Intelligence Group (AI group) is «to extend the way computers can interact with the real world: specifically to develop better sensory and motor equipment, and programs to control them.» {Minsky, Status Report II}. From such efforts, a robot or mechanical manipulator has emerged, consisting of a PDP-6 computer, an image dissector camera, mechanical arm and hand (see pictures).

AUG · 68

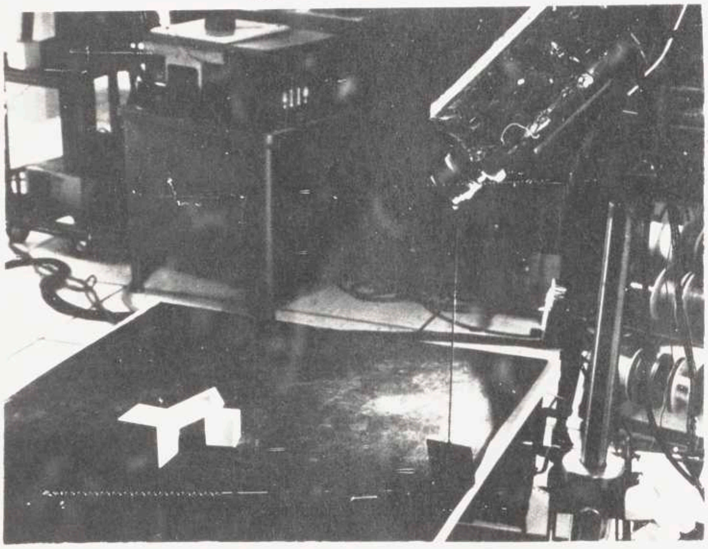
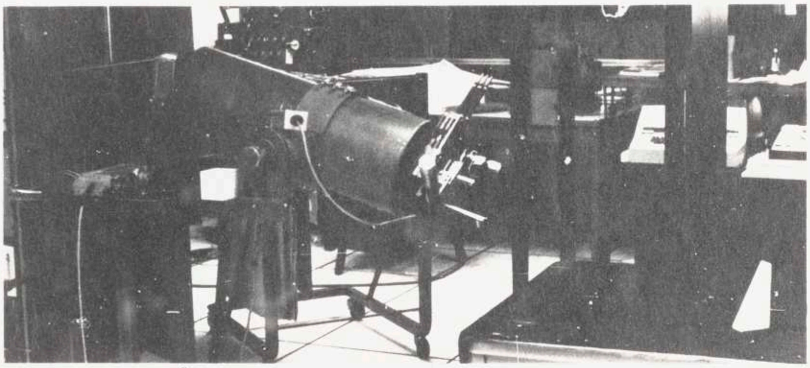


IMAGE DISSECTOR CAMERA

«These "eyes and hands" are eventually to be able to do reasonably intelligent things but first, of course, it is difficult enough to get them to do things that are easy for people to do.» {Ibid.}

An image dissector watches silently a triangular prism in the vision laboratory of the A.I. Group.



The work was naturally divided into visual information processing (computer vision) and manipulation and control of the arm-hand. Thus, when I came as a graduate student from the Politécnico de México to M. I. T. (Sept. 65) and became associated with the AI Group, I found a great interest there in graphical communication with computers. More over, it was felt that symbol manipulation techniques would be r elevant to this area. I was fortunate enough to have had some contact with the LISP language through some of its implementations: MB - LISP {McIntosh 1963} * and Hawkinson-Yates- LISP {Hawkinson 64}* at the Centro Nacional de Cálculo of the Politécnico; in fact, I became interested in the area because I felt that it would be possible to handle two-dimensional structures much in the same fashion as one handles lists (that is, one-dimensional structures or strings of symbols) in a pattern-driven language, such as CONVERT {1965}, recently finished at that time.

The area also offered a good opportunity to understand and evaluate several techniques, computers, equipment, etc. Consequently I decided to work in it.

(*) The parentheses { } always indicate a reference to the bibliography at the end of this thesis, where the complete title, date, etc., of the paper can be found.

SIMPLIFIED VIEW OF SCENE ANALYSIS

TO THE BUSY READER

This section presents a general view of the problems in the thesis and their solutions; if you are short of time,

(1) Read the abstract and this section.

(2) Choose some scenes from section 'Analysis of many scenes', and observe how the computer perceives them.

(3) Through the table of contents, select additional topics.

Scene Analysis Scene analysis is the result of interaction between optical data coming from the Eye, and knowledge about the visual world stored in the programs. In all that follows, the optical data entering through the Eye is reduced to a line drawing; this pass is called pre-processing, and it will be only briefly sketched here.

After preprocessing, such a line drawing is analyzed in order to discover and recognize given objects in it. The process is called recognition.

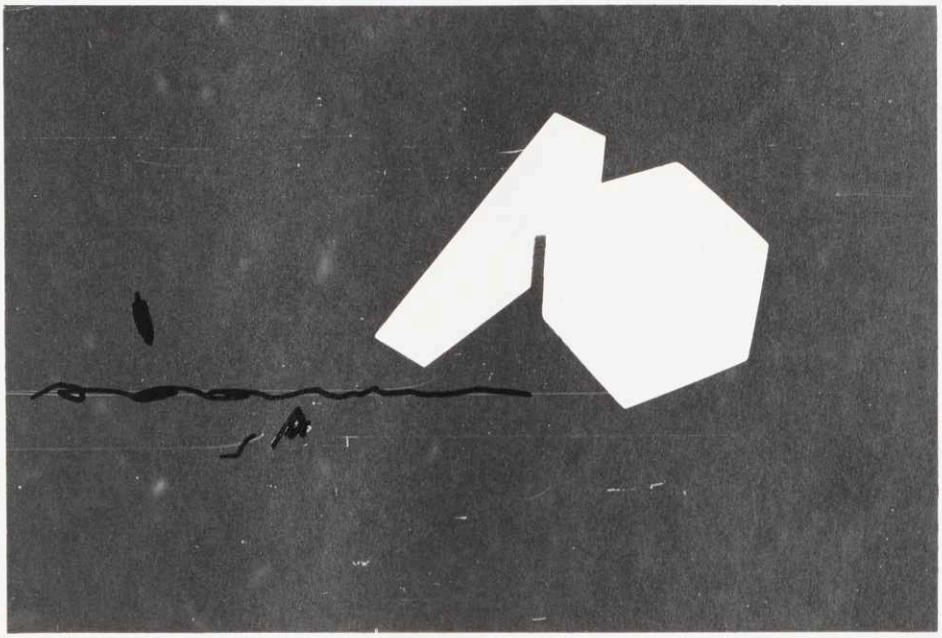
This thesis is concerned with recognition.

> The stylized presentation that follows is only an example; in particular, scene analysis does not need to follow the sequence pre-processing → recognition.
> See 'Division of work in Computer Vision' in page 60.

We now give a simplified exposition of both processes. Recognition will be discussed abundantly in the remainder of this thesis, since it is the main topic; readers who wish for more information on pre-processing or other approaches should consult the references, for instance {my MS Thesis} and {A C Shaw FJCC 68}. See also page 60.

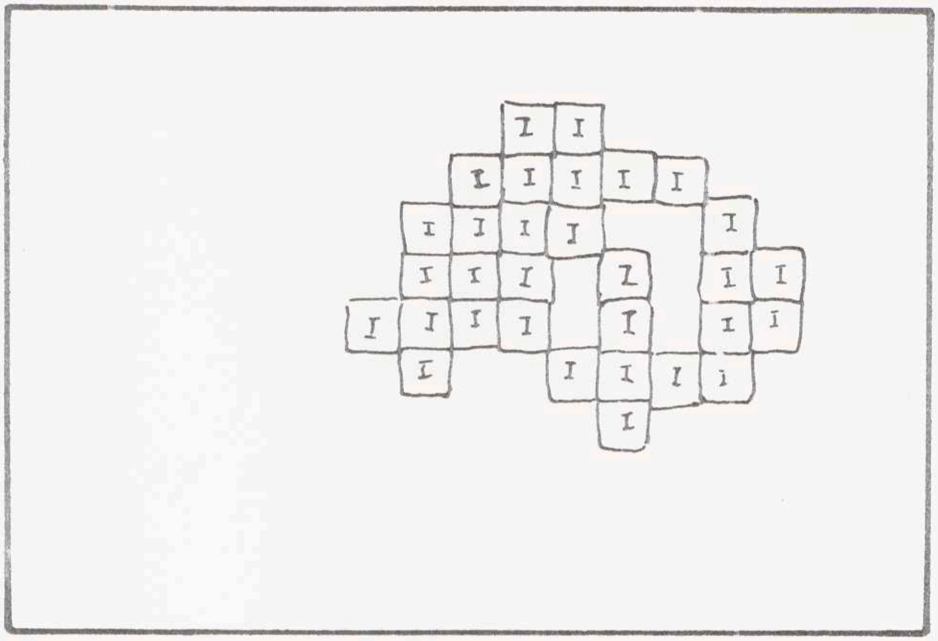
The picture shows a scene containing two light objects on a dark table.

AUG • 68

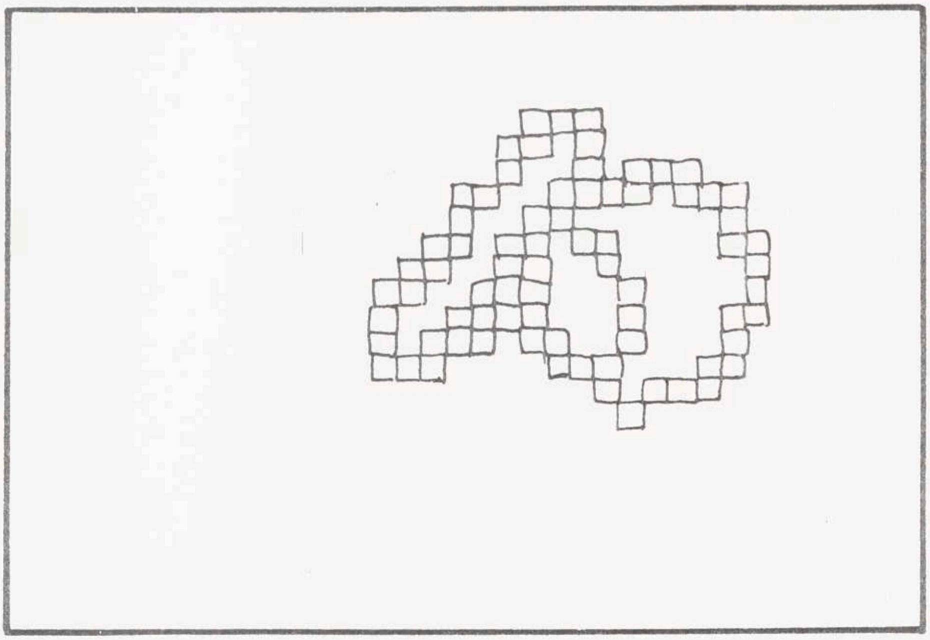


This scene could be entered into memory as a bi-dimensional array of numbers (logarithms of light intensity, to make the programs insensitive to changes in general light level); since our camera is a random-access device [each point (x, y) on the visual field can be accessed in roughly the same time], we prefer to read the point each time we want to know the light level at it, instead of reading it once at the beginning and keeping the numbers in memory. In one way or another, we effectively have a large bi-dimensional array of numbers representing our scene.

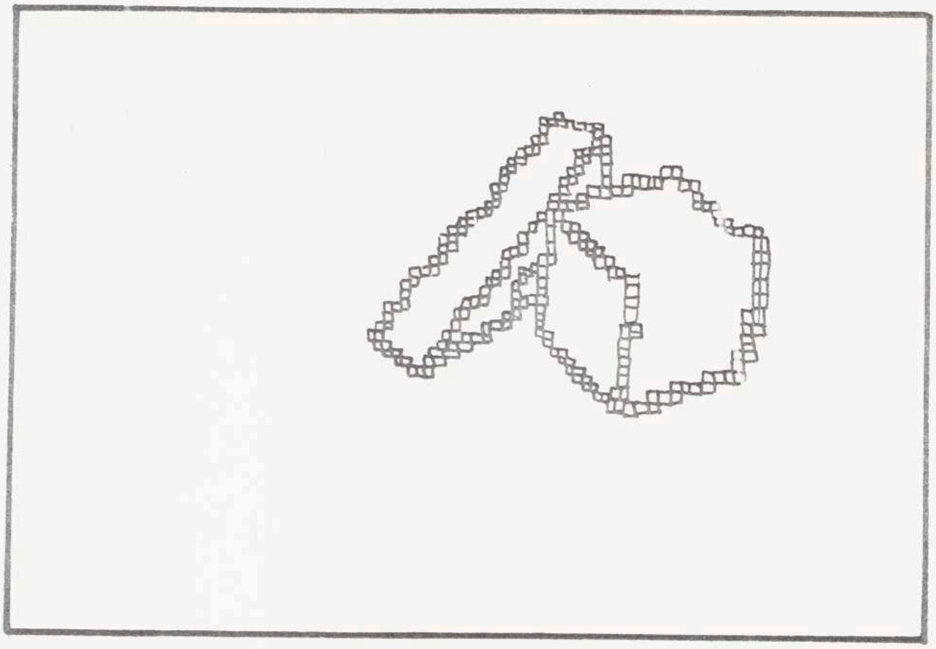
A coarse grid of about twenty squares per side is laid on top of the array (but see small box in previous page), and the program ignores the "homogeneous" squares --those squares whose corners have nearly the same values.

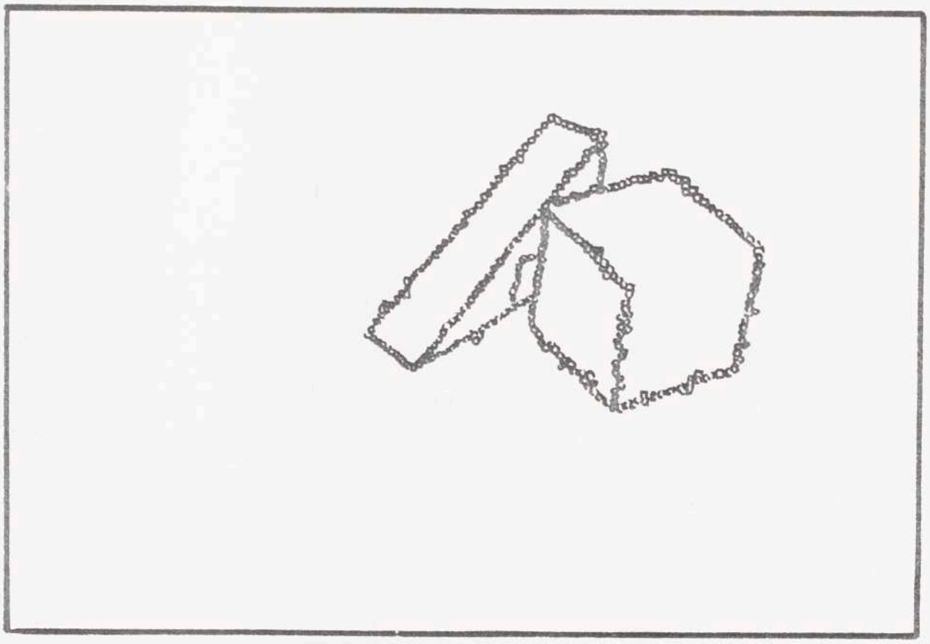


Each inhomogeneous square [I] is divided in four ⊞ , ignoring again the homogeneous sub-squares.



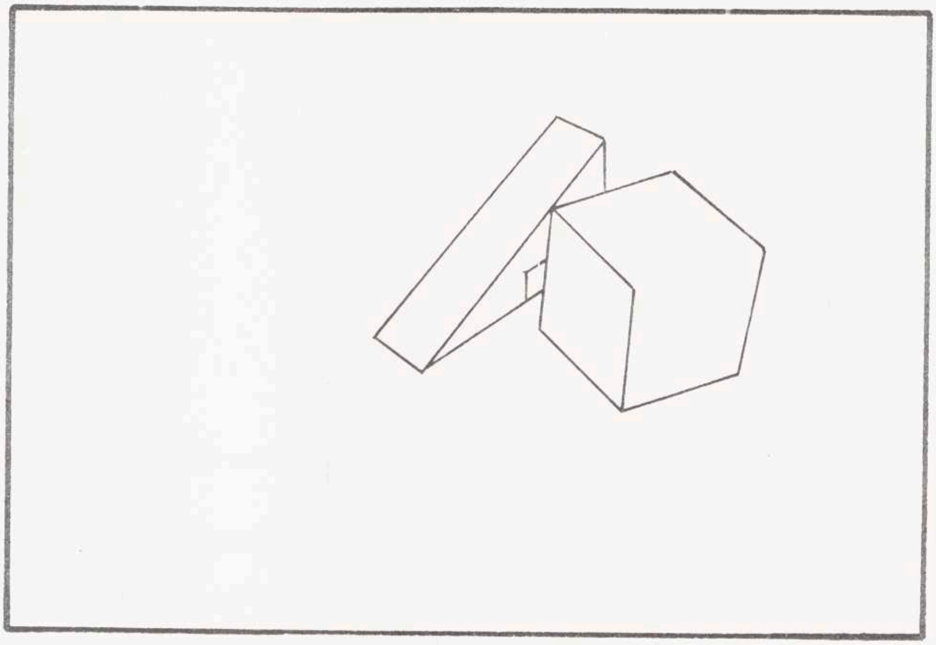
The process is repeated a few times more.





The squares are now reduced to lines and vertices.

The resulting analysis gives us the first chance to start working abstractly now, instead of continuing in "picture-point space." Preprocessing has finished.

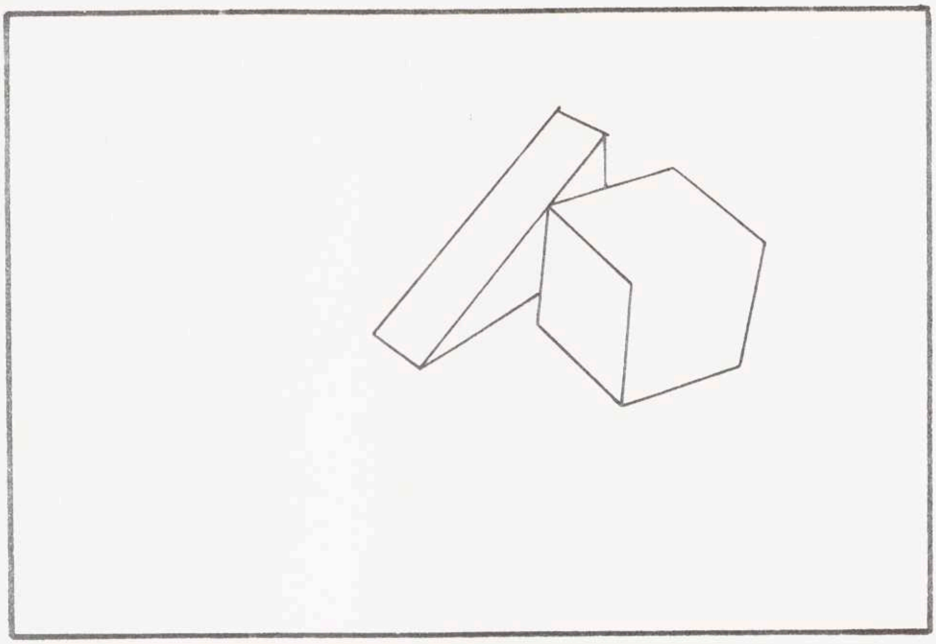


This and the next page describe proposed, but still unfinished, parts of the system.

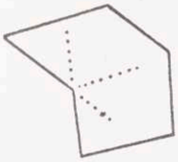
Recognition

The theme of this work is "Computer Recognition of Three-Dimensional Objects in a Visual Scene;" what follows is, therefore, merely a brief summary. (some of the described parts are proposed but unfinished). A more systematic presentation and classification of processes in recognition is found in 'Division of work in Computer Vision', page 60 in the section 'SEE, a program that finds bodies in a scene'.

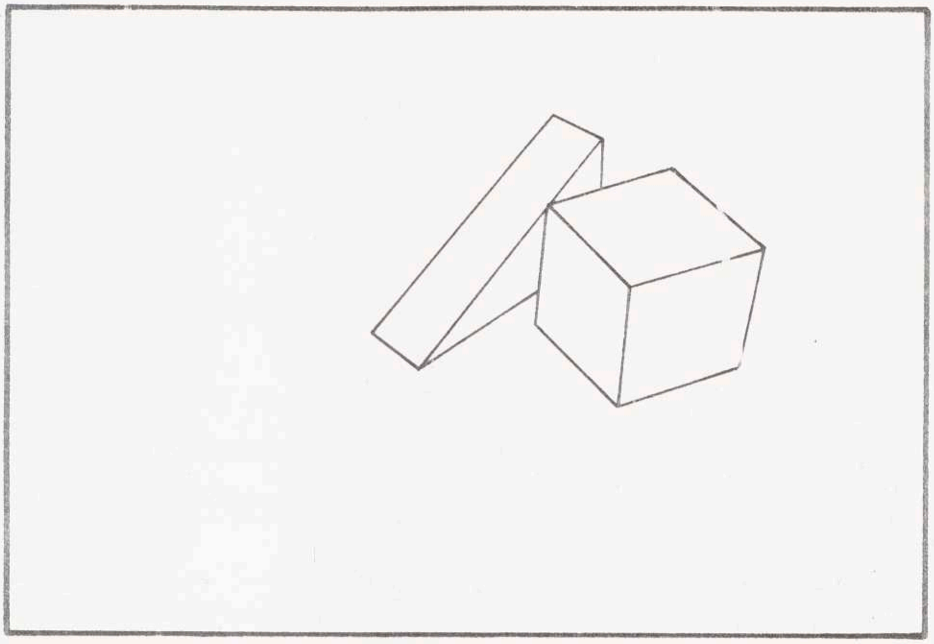
A program checks in the original scene, in both sides of each line, for continuation across the line, of textures, local cracks, etc. On these and other grounds, shadows are picked up and erased:



A line-proposer program studies the abstract or "symbolic" scene and, using some heuristics and general principles, proposes places where it is quite probable that a line is missing:



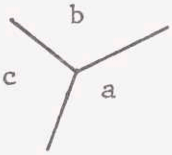
These places are searched by a line-verifying program, which is an specially sensitive test that uses fine measurements from the original scene, and often it will pick up a boundary that was missed in the less-intelligent homogeneity phase. Here it can be practical to apply a very strict and sensitive test, because the program knows very accurately where the line should be, if it really exists at all. For example, even if the two faces have almost equal illumination the Eye can pick up a thin, faint highlight from the edge of the cube. It would have been hopelessly expensive to look for such detailed phenomena over the wole picture at the start.



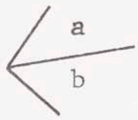
At this stage our program SEE (see Section (page 53) 4) comes into action. This program treats different kinds of local configurations as providing different degrees of evidence for 'linking' the faces. This evidence is obtained mainly at vertices, and at boundaries between regions.

A vertex is in general a point of intersection of two or more boundaries of regions. These regions might or might not be faces of a single body. SEE examines the configuration of lines meeting at the vertex to obtain evidence relevant to whether the regions involved belong to some object.

For instance, in the vertex configurations "ARROW" and "FORK"(a complete classification of vertices can be found below in table 'VERTICES'),

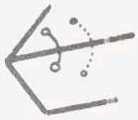


"FORK"

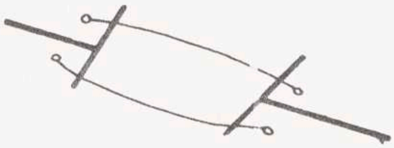


"ARROW"

the "FORK" suggests linking face a to face b, b to c, c to a. The "ARROW" links a with b. A "leg" (which depends on nearly parallel lines) would add a weak link, in addition to the ordinary

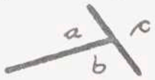


'LEG'
(Weak link shown dotted)

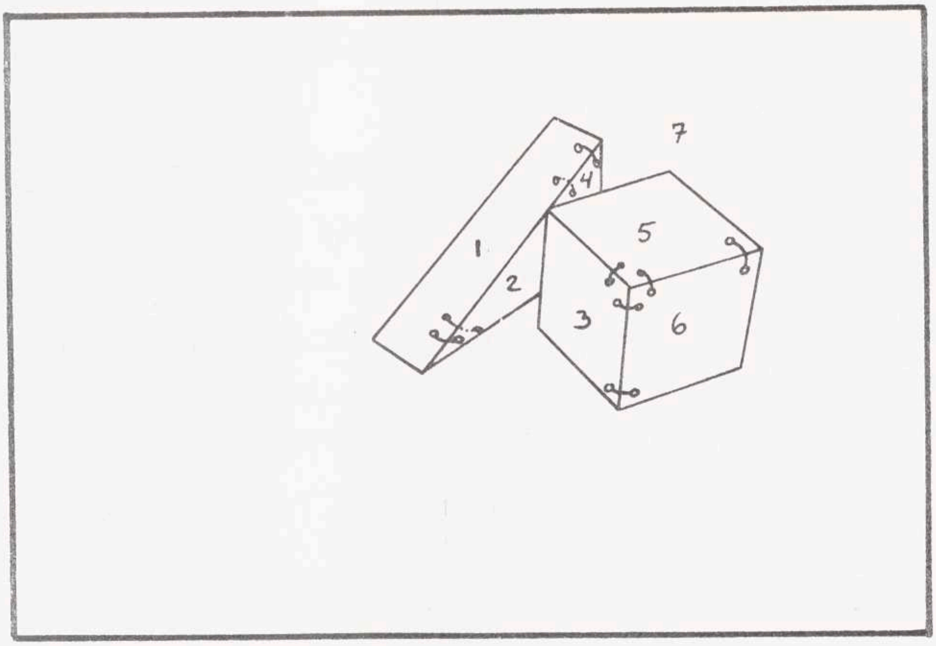


Matching T's.
(two strong links)

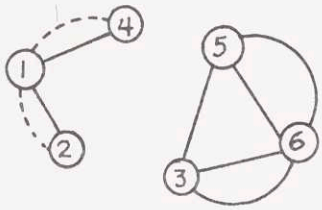
(or strong) link placed by its 'arrow'; a "T" looks for a matching "T", and if found, two strong links are placed as shown. Also, a "T" counts against (inhibiting, that is) linking a with c, or b with c.



These links, for our example, are



and may be represented as



[weak links are dotted]

indicating two groups of linked faces, that is, two bodies:

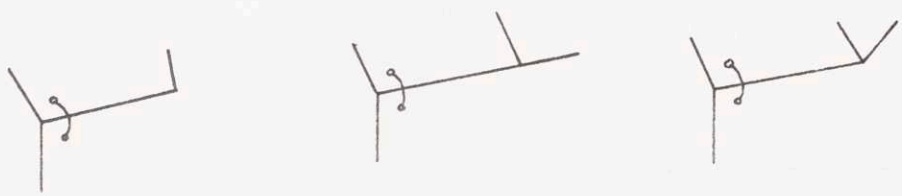
```
(BODY 1. IS 1 2 4)
(BODY 2. IS 3 5 6)

     If in addition we give at this point to
     the computer the definition or concept
     of a 'triangular prism', through an ab-
     stract model of it {my MS Thesis}, we
     can get

(1 2 4 IS A TRIANGULAR-PRISM)
(3 5 6 IS A CUBE)
```

Recognition has finished.

Analysis of several examples

A larger variety of kinds of evidence is used in more complicated scenes, making the program more intelligent in its answers:

(1) The links themselves are inhibited by conditions or configurations at the neighbor vertices and faces; for instance, in the case of a "FORK", the (strong) links indicated below are inhibited:



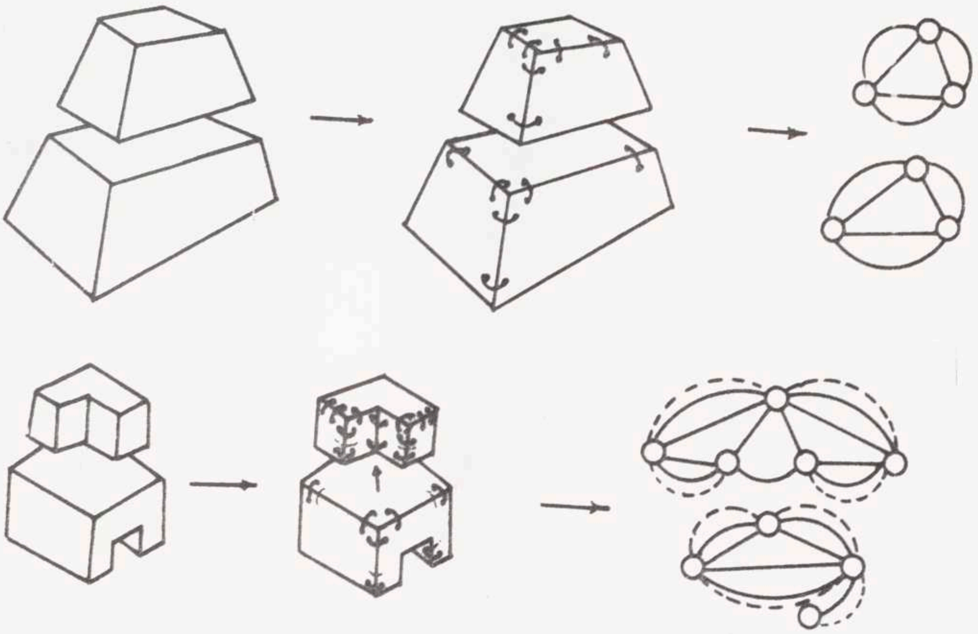
(2) The links to the background are ignored [complete descriptions of conditions for producing and cancelling links are to be found in section 'SEE, a program that finds bodies in a scene'].

(3) A hierarchical scheme is used that first finds subsets of faces that are very tightly linked (e. g., by two or more links).

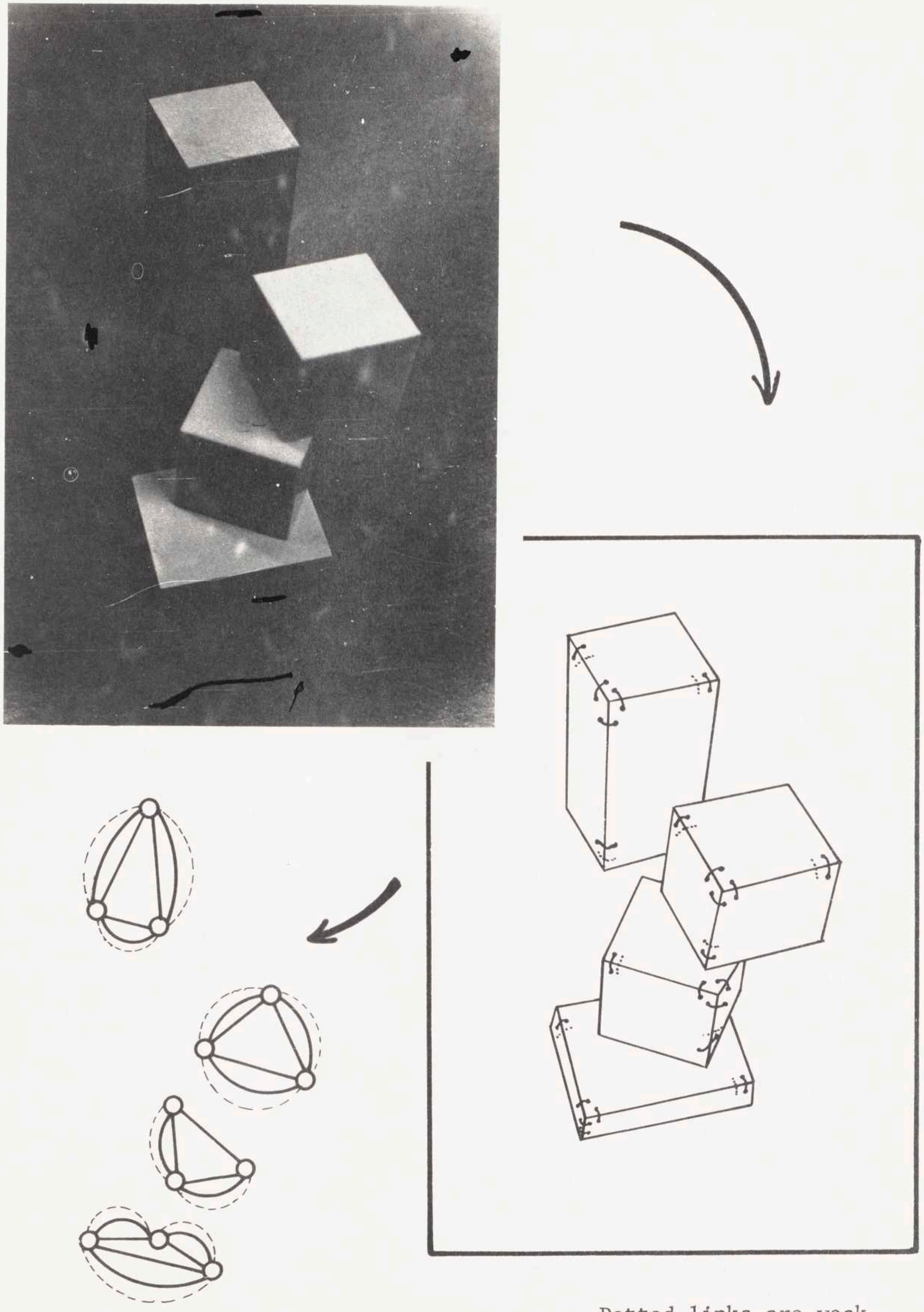
These "nuclei" then compete for more loosely linked faces (faces linked through one weak link and one strong link ,
or one face completely unlinked, except by one strong link).

By not considering a single link, weak or strong, as enough evidence for assigning two faces as part of the same object, this algorithm requires two "mistakes" (that is, two careless placements of links between regions that should not be considered as forming the same body) to make an identification error.

The bodies of the following scenes are found by SEE without difficulty.

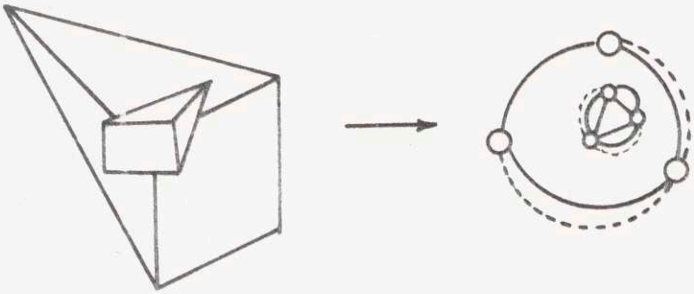


Note that of the strong links available to the "FORK" marked with an arrow, two were prohibited or inhibited and only one is produced by SEE.



Dotted links are weak.

In the following figure, the "FORK" of the big object is missing.



Statement of Rules We will re-state the rules under (3) of page 22.

Region (definition). Surface bounded by simply closed curves. We will consider the outer background (:16 in fig 'L10', page 59) to be also a region.

Nucleus (definition). A nucleus (of a body) is a set of regions.

Linked nuclei (definition). Two nuclei A and B are linked if regions a and b are linked where $a \in A$ and $b \in B$.

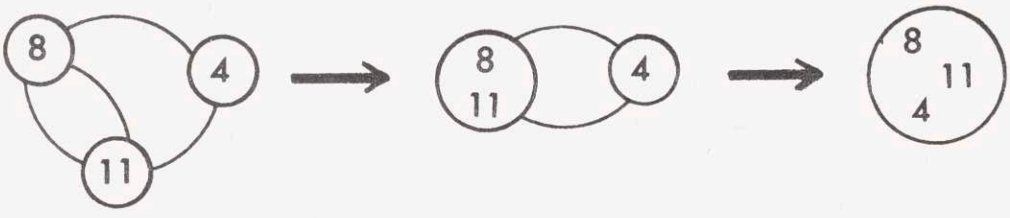
First rule: If two nuclei are linked by two or more strong links, they are merged into a larger nucleus.

For instance, regions :8 and :11 are put together, because there

exist two strong links among them, to form the nucleus :8-11.

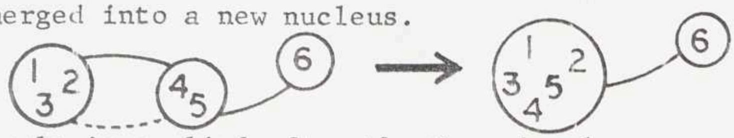
Maximal nuclei: Starting from nuclei containing individual regions, we let the nuclei grow and merge under the First rule, until no new nuclei can be formed. When this is the case, the scene has been partitioned into several "maximal" nuclei; between any two of these there is at most one strong link.

For instance, regions :8 and :11 are put together by the First rule; now we see that region :4 has two links with nucleus :8-11, and therefore the new nucleus :8-11-14 is formed. This last is a maximal nucleus.



The First rule is applied again and again, until all nuclei are maximal nuclei; then the following rule is applied:

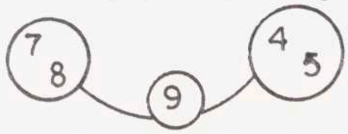
<u>Second Rule</u>: If nuclei A and B are joined by a strong and a weak link they are merged into a new nucleus.



The Third rule is applied after the Second rule.

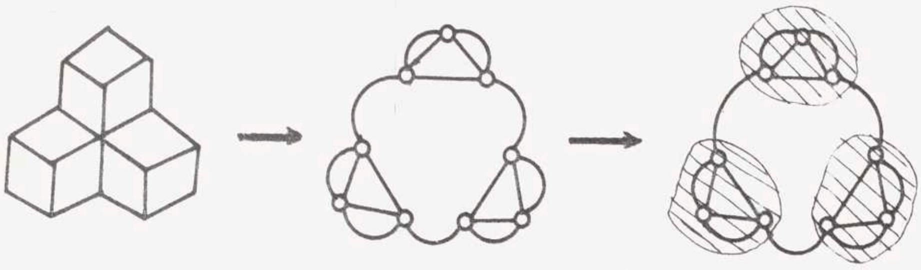
<u>Third Rule</u>: If nucleus A consists of a single region, has one link with nucleus B and no links with any other nucleus, A and B are merged.

(10 11) does not join the bigger nucleus because (10 11) does not consist of a single region. Below, 9 does not join (7 8) or (4 5) because 9 has two links:



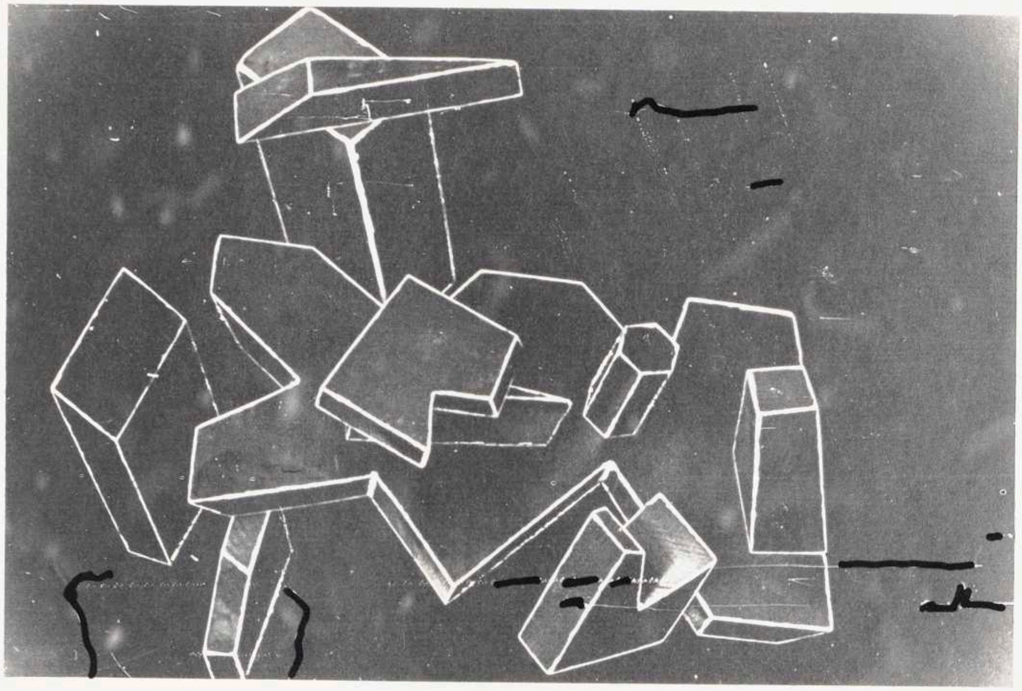
The Third rule tends to avoid proposing bodies consisting of a single region.

The next example shows how three "false" links failed to lead SEE into error:

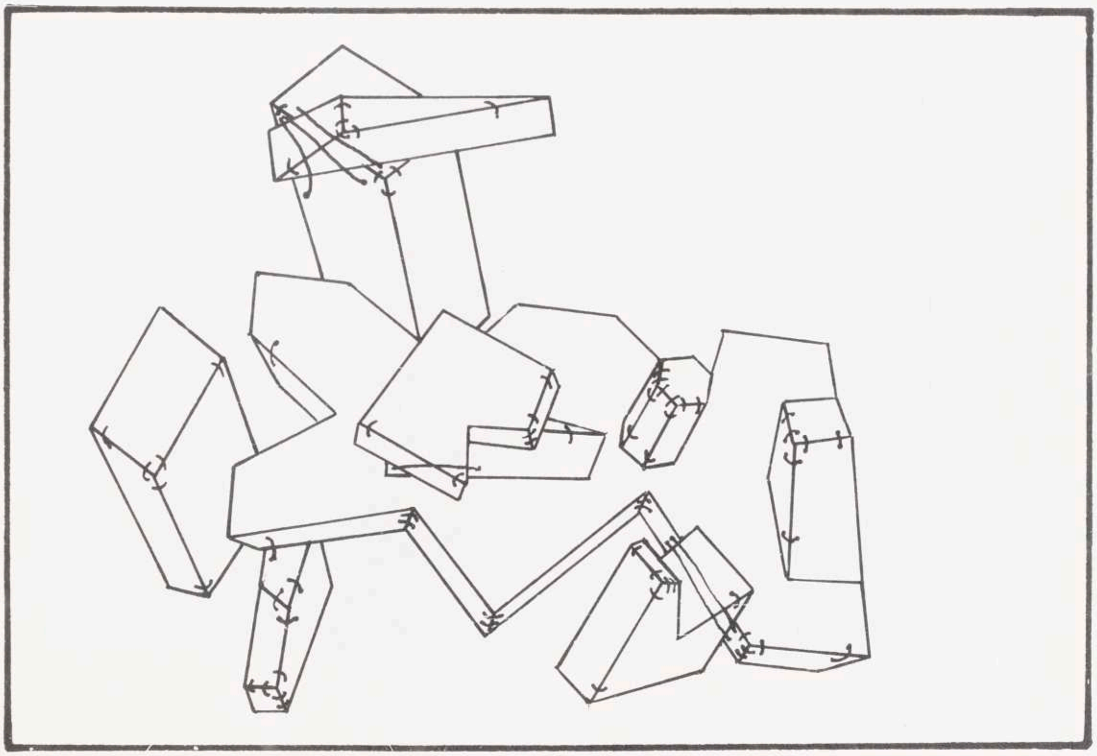


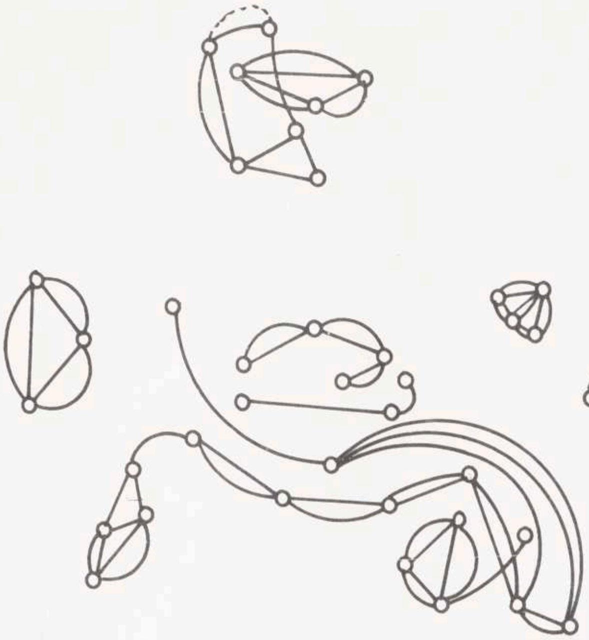
Here three links were erroneously placed but SEE did not get confused by them.

In complicated scenes, coincidences cause two objects to line up. As a result, vertices of different objects are merged, two objectively different lines appear as one and so on. The next example illustrates these phenomena and shows how SEE copes with the problem.



SEE transforms the above scene as follows:





As we see, the nuclei are going to be correctly formed, and SEE will also analyze this scene correctly.

The bodies do not need to be rectangular, prismatic, convex. They only need to be rectilinear. As we will see later, even curve objects may be identified, under certain restrictions (cf. Table 'ASSUMPTIONS').

AUG · 68

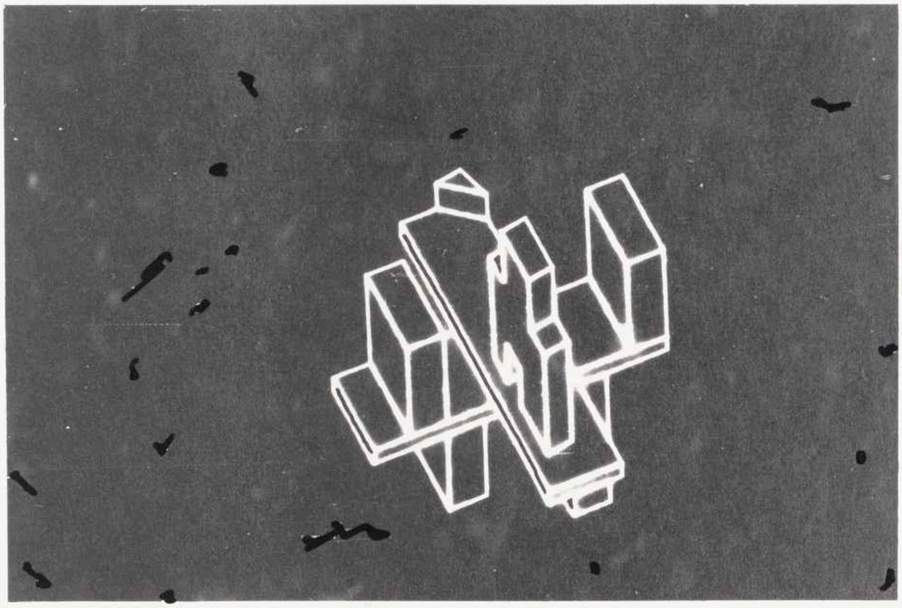
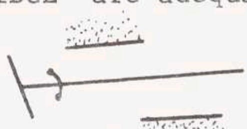


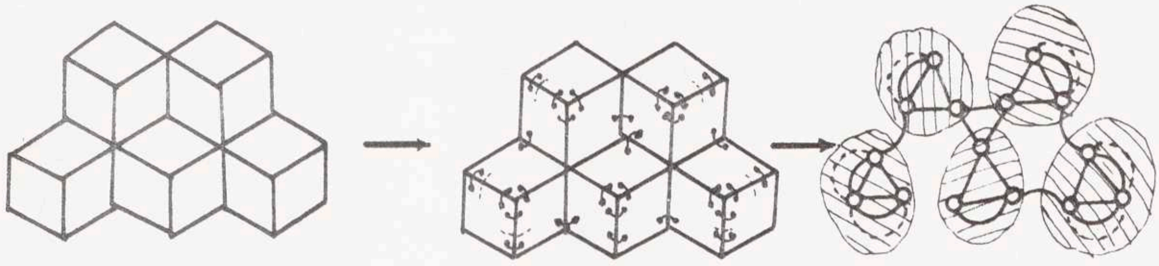
Figure 'BRIDGE'

All the bodies in "BRIDGE" are adequately found. A new heuristic is used here:



three parallel lines comprising regions that are not background, and having the background as a neighbor, and a 'T' in the center line, originate a strong link, as shown above.

The following locally ambiguous scene is correctly parsed by our program:



If we add another block to the right, the program makes a mistake and fails to see one of the inner cubes:

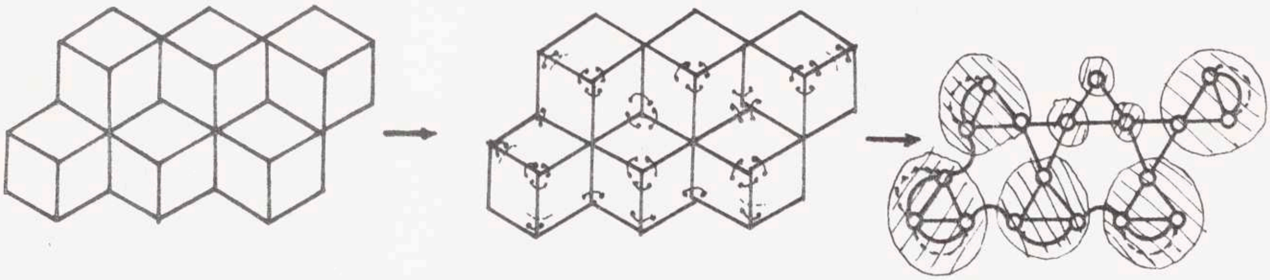


Figure 'MOMO' also gets decomposed accurately:

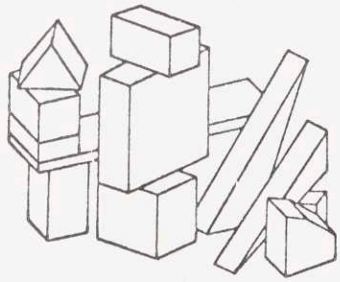
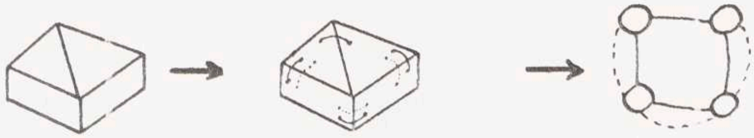
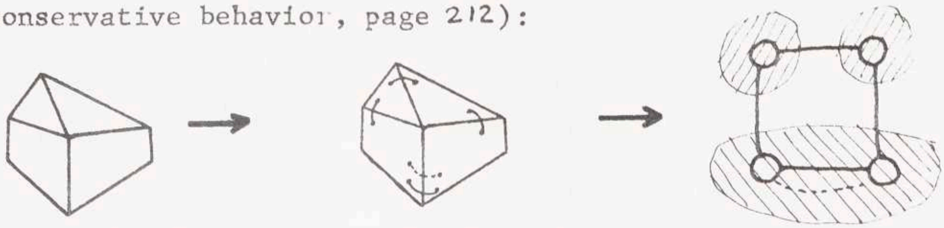


Figure 'MOMO'

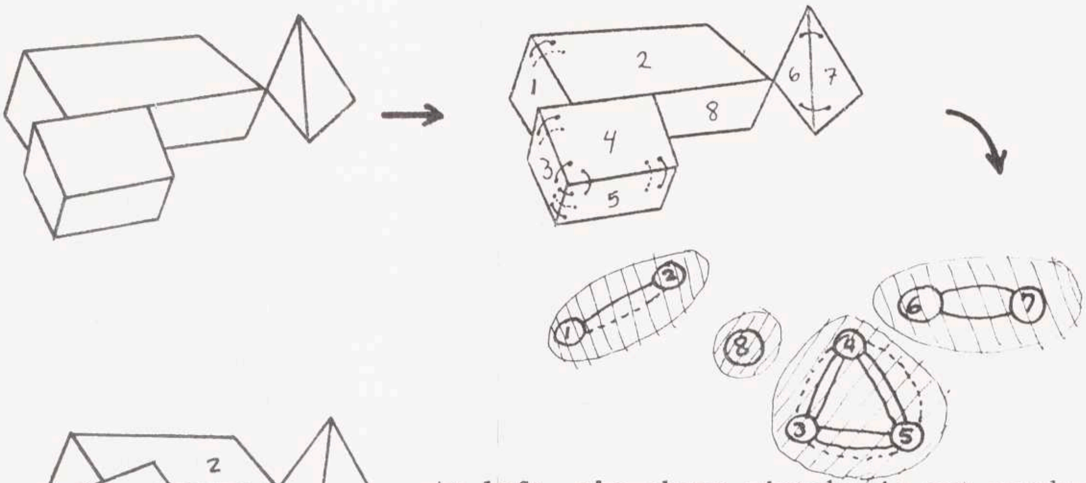
The local links allow correct identification of the following body:

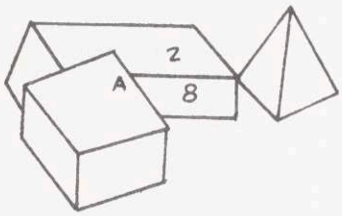


If the lateral faces do not have parallel edges, a mistake occurs (conservative behavior, page 212):



Another mistake occurs in the following scene:





At left, the above mistake is not produced because vertex A links :2 and :8, by the new heuristic introduced in 'BRIDGE'.

Conclusion

The performance of this program shows that it is possible to separate a scene into the objects forming it, without needing to know the objects in detail; 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 program will be fully analyzed in the following pages.

Problems in analyzing a visual scene *

The problem of taking a two-dimensional image (or several such images), and constructing from it a three-dimensional interpretation, involves many operations that have never been studied, to say nothing of being realized on a computer. We will list some of these here; a more complete list is found in my M.S. Thesis {MAC TR 37}; some have been side-stepped or ignored by the present recognition system; the problems which we did solve are discussed through the text.

Among the facilities that must be available are:

a) Spatial frame-of-reference: setting up a model of the relation between the eye(s) and the general framework of the physical task, i. e., where are the background, the "table" or working surface, and the mechanical hand(s)?

b) Finding visual objects, and localizing them in space with respect to the eye-table-background-hand model.

c) Recognizing or describing the objects seen, regardless of their position, accounting for partly-hidden objects, recognizing objects already "known" by descriptions in memory and representing the three-dimensional form of new objects.

d) Building an internal "structural model" of what has been seen, for the purpose of task-goal analysis.

Among the important factors are the effects of:

1. Both the camera's focus and its depth-of-focus.
2. Illumination of the objects. Light affects the appearance of objects in obvious and subtle ways -- in scenes with multiple objects and lights we get complicated shadows, which have to be detected or rejected. The boundary between two faces may disappear if they get equal illumination from a diffuse light source.
3. Perspective and distance effects. Even for geometric objects with flat surfaces, the two-dimensional projection of their surface

---

* Adapted from Status Report II {Minsky 67}. See also Project MAC Progress Report {1967, 1968}.

features can take many forms, and the system has to be able to deal with all of them. It works both ways, of course: once identified, the appearance can give valuable information about the object's orientation, size, and even (under some conditions) its absolute spatial locations {Roberts 1963}.

4. Accidental vs. essential visual features. Two objects of the same shape and location can have very different visual presentations because of their surface textures and markings. We need to distinguish these two-dimensional "decorations" from real three-dimensional spatial features.

Other projects

Here are the main robot groups at a panel discussion.

1968 fall joint computer conference
DECEMBER 9-10-11 san francisco civic center

tuesday morning

december 10

8:45 a.m. / nourse

Chairman:
DR. BERTRAM RAPHAEL
Stanford Research Institute
Menlo Park, California

problems in the implementation of intelligent robots

This session, the second of three sessions on robotry, will consist of a panel discussion among technical people involved in the design and construction of mechanical devices that are capable of significant independent "intelligent" behavior, usually by means of computer control. The projects represented on this panel have drawn upon state-of-the-art capabilities in many technologies including mechanical engineering, pattern recognition, heuristic programming, neural networks and computer systems. Thus, the discussion which will be conducted at a fairly technical level should be of interest to engineers and scientists concerned with the problems of interfacing a variety of disciplines, as well as to those interested in learning about the nature of current embryonic "robot" systems.
NOTE: Tickets priced at $5.00 each (including lunch) for the all-day tour of "live robot" installations on Wednesday, Dec. 11th, will be available at this session.

Panel Members

MR. L. CHAITIN
Artificial Intelligence Group
Stanford Research Institute
ROBOT STUDIES AT STANFORD RESEARCH INSTITUTE

PROF. J. A. FELDMAN
Computer Science Department
Stanford University
THE ROBOT PROJECT AT STANFORD UNIVERSITY

DR. T. SHERIDAN
Dept. of Mechanical Engineering
MIT
HUMAN CONTROL OF REMOTE COMPUTER MANIPULATORS

MR. R. J. LEE
Air Force Avionics Lab.
Wright–Patterson AFB
GENERAL PURPOSE MAN-LIKE ROBOTS

PROF. S. PAPERT
Artificial Intelligence Project
MIT, Project MAC
THE MIT HAND-EYE PROJECT

MR. L. SUTRO
Dept. Aeronautics and Astronautics
MIT
ROBOT DEVELOPMENT AT THE MIT INSTRUMENTATION LABORATORY

RELATED RESEARCH

Previous work by the author

CONVERT

A programming language is described which is applicable to problems conveniently described by transformation rules. By this is meant that patterns may be prescribed, each being associated with a skeleton, so that a series of such pairs may be searched until a pattern is found which matches an expression to be transformed. The conditions for a match are governed by a code which also allows subexpressions to be identified and eventually substituted into the corresponding skeleton. The primitive patterns and primitive skeletons are described, as well as the principles which allow their elaboration into more complicated patterns and skeletons. The advantages of the language are that it allows one to apply transformation rules to lists and arrays as easily as strings, that both patterns and skeletons may be defined recursively, and that as a consequence programs may be stated quite concisely.

Abstract of Convert paper in Comm. A.C.M.

Because it is easy to write and modify a program in Convert, the language has been extensely used to quickly test 'good' and "great" ideas, new algorithms, etc. It is embedded in the LISP of the PDP-6 computer (A.I. Group), in the IBM-7094 (Pr. MAC, MIT); in the CDC-3600 (Uppsala University, Sweden), in the SDS-940 (Univ. of California, Berkeley). A paper in the A. C. M. and {MAC M 305} describe the language; examples of simple programs written in Convert are in {MAC M 346}; a book article {Patterns and Skeletons in Convert} is oriented toward the Lisp consumers. For our Spanish readers, two Bachelor's Theses {Guzmán 1965} {Segovia 1967} describe the language and processors, and give examples.

SCENE ANALYSIS

(1) Polybrick {MAC M 308} {Hawaii 69} is a Convert program that works on a scene or picture, expressed as a line drawing, and finds parallelepipeds in it.

(2) We would like to be able to specify in some suitable notation <u>models</u> of the classes of objects we are interested in (such as 'cube', 'triangular prism', 'chair'), and make a program look for all instances of any given model in a given scene or figure. Two arguments would have to be supplied to our program: the <u>model</u> of the object we are interested in, and the <u>scene</u> that we want to analyze. Programs to do this are described in {AFCRL-67-0133} and {MAC M 342}. In these early programs, partially occluded objects get incorrectly identified. These programs are also written in Convert, and work by transforming or compiling the model, written in a picture description language, into a Convert pattern, which searches the scene for instances of the model.

(3) A Master's Thesis {MAC TR 37} discusses many ways to identify objects of known forms. Different kinds of models and their properties are analyzed.

(4) It is important to be able to find the bodies that form a scene, without knowing their exact description or model. SEE is a program that works on a scene presumably composed of three-dimensional rectilinear objects, and analyzes the scene into a composition of three-dimensional objects. Partially occluded objects are usually properly handled. This program was discussed in {MAC M 357}, {Guzmán FJCC 68} and {Pisa 68}, and this thesis discusses a later version.

(5) The present thesis goes beyond these topics to discuss also handling of stereo information (two views, left and right, of the same scene), improvements to deal with noisy (imperfect) input, figure-background discrimination, and a few other subjects.

<u>Canaday</u>

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

The problem of machine recognition of pictorial data has long been a challenging goal, but has seldom been attempted with anything more complex than alphabetic characters. Many people have felt that research on character recognition would be a first step, leading the way to a more general pattern recognition system. However, the multitudinous attempts at character recognition, including my own, have not led very far. The reason, I feel, is that the study of abstract, two-dimensional forms leads us away from, not toward, the techniques necessary for the recognition of three-dimensional objects. The perception of solid objects is a process which can be based on the properties of three-dimensional transformations and the laws of nature. By carefully utilizing these properties, a procedure has been developed which not only identifies objects, but also determines their orientation and position in space.

Three main processes have been developed and programed in this report. The input process produces a line drawing from a photograph. Then the three-dimensional construction program produces a three-dimensional object list from the line drawing. When this is completed, the three-dimensional display program can produce a two-dimensional projection of the objects from any point of view. Of these processes, the input program is the most restrictive, whereas the two-dimensional to three-dimensional and three-dimensional to two-dimensional programs are capable of handling almost any array of planar-surfaced objects. {from Roberts}

Roberts 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.

Mechanical Manipulator Groups

(see also page 32).

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.

THE CONCEPT OF A BODY

In this section definitions of a body or object will be proposed. The criterion is that they agree in general with the common use of the word 'body', while at the same time they should lead themselves to implementation into a computer program.

Introduction

Our ultimate interest is to examine a two-dimensional scene (a picture, line drawing, or painting), presumably a representation (projection, photograph) of a three-dimensional scene (a subset of the "universe" or "real world") and to find in it objects or bodies contained in the real scene. More specifically, the aim is to find the two-dimensional representations (projections, photographs) of the different three-dimensional bodies present in the scene.

> The phrase "two-dimensional representation of a three-dimensional body" will be shortened to "two-dimensional body" or even to "body", when no confusion arises.

That is, we have to analyze a two-dimensional scene into collections of two-dimensional entities (surfaces, regions, lines), each of which makes "three-dimensional sense" as a two-dimensional projection of a three-dimensional body.

The problem is inherently ambiguous

A scene can be considered as a set of surfaces (faces or regions), a body belonging to that scene is then an "appropiate" subset of this collection. Therefore, the problem of finding bodies in a scene is equivalent to the problem of partitioning the set into appropiate subsets, each one of them representing or forming a body (scene "CHURCH").

The problem is inherently ambiguous, since different collections of three-dimensional bodies can produce the same 2-dim scene, therefore a given scene can be partitioned in many ways into bodies (without such a requirement, the problem has a trivial solution, in which each region is a projection of a separate, perhaps prismatic, body [cf. also

Metatheorem of page 39]).

It is desired to make a "natural" partition or decomposition of the scene, natural in the sense that will agree with human opinion.

To define a <u>three-dimensional</u> body is no problem [a philosopher may disagree, perhaps in singular cases]:

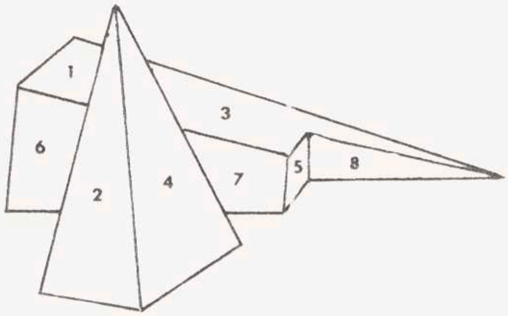


Figure 'CHURCH'

Set of eight elements. Adequate subsets (bodies) are [2 4], [1 3 5 6 7 8]. In a more complicated example, people may differ in their parsing of scenes.

<u>Three-dimensional body</u> (definition): A connected volume limited by a continuous, two-sided surface composed of portions of planes.

Restriction: The above definition covers only polyhedral bodies, that is, those having flat faces.

Restriction: No holes.

No-restriction: Bodies do not need to be convex.

Roughly speaking, a three-dimensional body is something that does not fall apart into pieces when lifted [this may be used as an operational definition of a body, given a mechanical manipulator to make the necessary tests].

Given a three-dimensional body, we generate a two-dimensional body by taking a picture of it, as follows.

<u>Two-dimensional body</u> (definition). Figure formed by the projection of a three-dimensional body. Generally, the projections is isometric or perspective.

Thus, this is a view in two dimensions of a solid body, from some particular point of view.

Unfortunately, a two-dimensional body could come in this way from any of several different 3-dim bodies or, what is worse, two 3-dim bodies together can give rise to a single 2-dim body. For instance, in fig. "BENT",

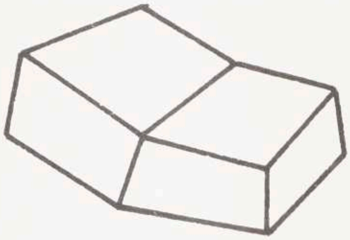


Figure 'B E N T'
Two blocks, or a bent brick.

this two-dimensional body could be generated by a "bent brick" or by two blocks adjacent to each other. We are dealing with one three-dimensional body in the first case, with two in the second. But the 2-dim entity (namely, the drawing of figure 'BENT') is the same, and we are confronted with an inherent ambiguity.

Sibelius' Monument A more striking example is given in Fig. 'SIBELIUS', which could be the representation of 365 cylindrical bodies, or the picture of a sculpture (one body) in Helsinki.

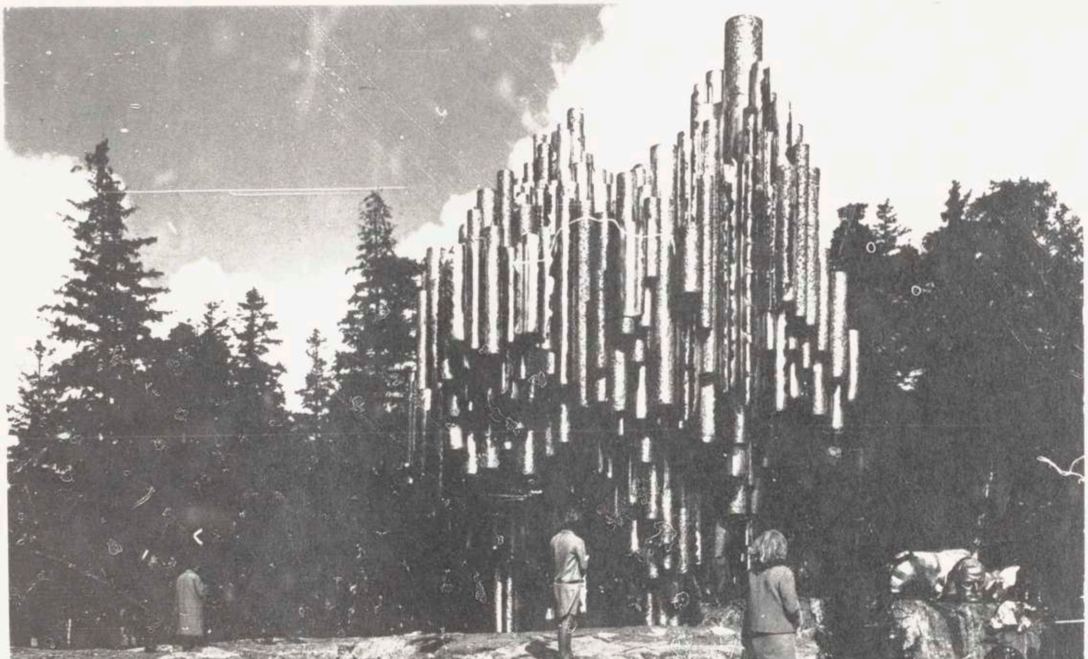


Figure 'SIBELIUS'

Such colorful contradictions point towards the need to lay down a more careful definition of our task. For instance, no one would think that figure 'CUBE'

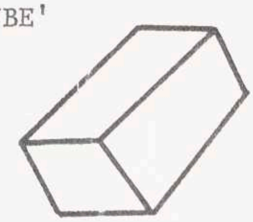
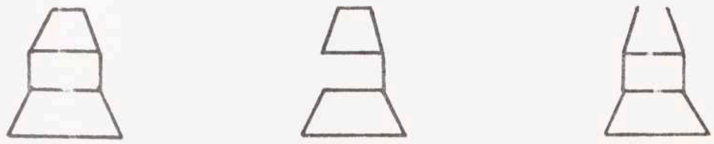


Fig. 'C U B E'
No one would think...

contains three bodies. Nevertheless (see fig. 'PARALLELEPIPED' in next page), that could be the case.

These two extremes are to be avoided by an appropiate definition of a body and the corresponding computer program.

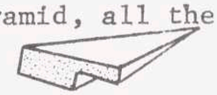
Legal scene That 2-dim scene in which each line is boundary of some region.



Legal scene. Illegal. Illegal.

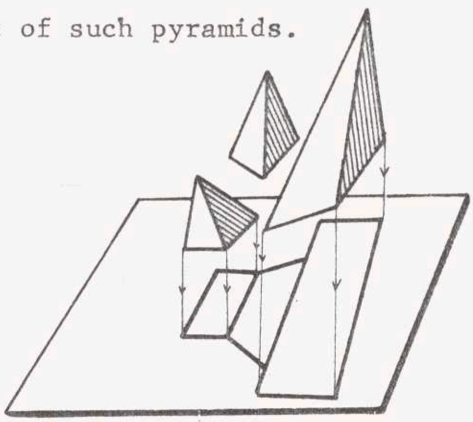
See also comments to scene R3, and 'Illegal Scenes' (page 217), in section 'On noisy input'.

Metatheorem Any legal scene can always be the projection of one or more three-dimensional objects.

To prove it, suffices to note that each legal scene is composed by regions  , and each of them could be interpreted as the basis of a pyramid, all the faces meeting at the cuspid occluded by the basis. 

Therefore, each legal scene can be obtained by projecting or photographing an adequate arrangement of such pyramids.

We can always construct a legal scene by photographing (or projecting) suitable 3-dim polyhedra.



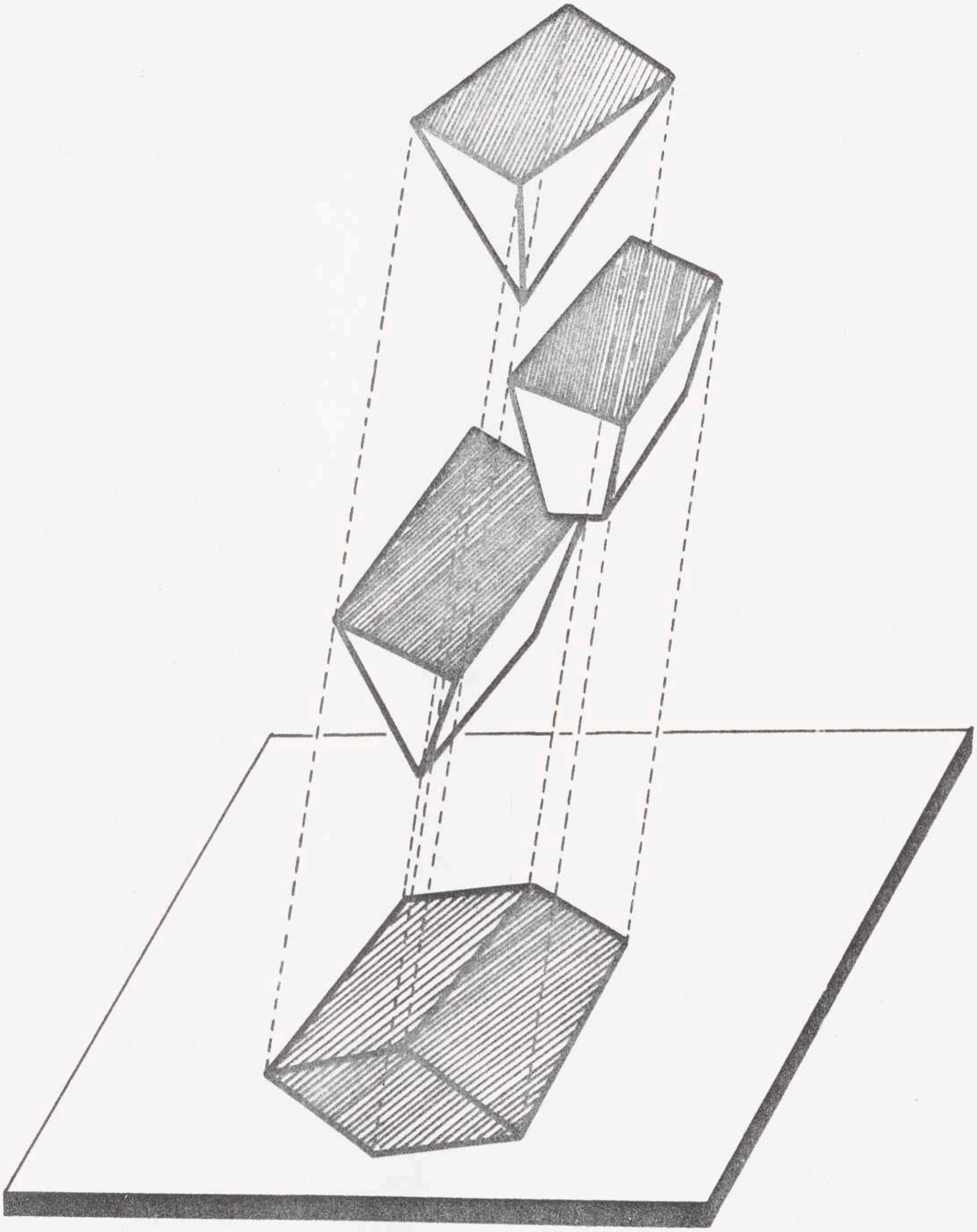


Figure 'P A R A L L E L E P I P E D'
An improbable decomposition of a scene.

Trivial partition By the use of the metatheorem, we can always find a decomposition of a visual scene into three-dimensional bodies; we call this answer "trivial". Humans do not split scenes this way. Our program should not, either.

But the metatheorem points out that "impossible scenes" are never found among the legal scenes (see section 'On Optical Illusions'); these always have at least one interpretation. [end of "Trivial partition"]

We are trying to give criteria for proposing bodies that will suit our ends, which are to define a "reasonable" or "standard" body. This will permit us to judge the performance of a program designed to find objects in a scene.

Several criteria are possible:

1. Roberts {1963} suggests: given several models of three-dimensional bodies, use some numerical techniques, such as least squares fitting, to find which model fits best through a suitable transformation, and accept this match if the error is tolerably small. Complicated compositions of elementaries bodies are considered.
2. Ledley {1962} would propose: in terms of suitable primitive components (arcs, legs, etc.), make a syntactical analysis of the scene, with the help of a grammar, in such a way that the models of the object you want to identify are formed recursively from these primitive components and (perhaps) other bodies. Narasimhan {1962} and Kirsch {1964} would agree on this linguistical approach. A. C. Shaw {Ph. D. Thesis} assents.
3. Guzmán {1967} suggests: prepare models which specify a fixed topology but where other relations (length of sides, parallelism of two lines, equality of angles) are specified through the use of open variables (UAR variables, in CONVERT). Evans {1968} would agree with that.

These approaches require the existence of a model which describes the object to be identified; the model specifies a particular 3-dim object (or a class of them). These approaches are answering more than what was asked; they tell not only "yes, it is a body", but also "it is a pyramid". The current question is more general. It is desired to know if something is a body, any body, even one which has not been seen before.

If it were possible to implement a program to answer that question, then that would be a working definition of a body. SEE is a program which comes close to this goal, so that it could be pragmatically stated:

<u>2-dim body "a la SEE"</u> (definition). A body is each set of regions recognized by the program SEE as such.

This definition allows the following

Criticism: A perfect way to hunt lions is to capture any entity E, and to call that a lion, by definition.

That is, although this definition is precise, SEE may make decisions "contrary to common sense"; also, for purposes of judging the behavior of the program, this definition is useless, since SEE will be perfect 100 per cent of the time, irrespective of its answers.

We are, finally, tempted to conclude that 'common sense', or better, "human common sense" plays a role in the definition of a body, since what we are trying to characterize is a <u>usual body</u>, <u>normal body</u>, <u>common body</u>, etc. But even people may differ in their parsings of scenes. We could, of course, give a scene (such as 'MOMO') to 100 subjects, ask them to identify the different bodies in it, and come up with some sort of 'average' or 'general consensus':

<u>2-dim body</u> (statistical and human-behavioral definition). Each one of the subsets into which a scene is partitioned by many subjects. It is understood that, in this spirit, the human objects should be motivated to satisfy a

<u>Simplicity criterion</u>: Of the several "reasonable" interpretations (decompositions) of a scene, the one which contains the smaller number of bodies is preferable.

That is, an explanation or decomposition is simpler (and preferable) if it can be done with fewer parts.

Simplicity is not to be achieved at any cost, since the parsing of the scene has to produce 'plausible' bodies, since "simplicity" could be always achieved if each scene is reported as a single, gigantic body, obtained perhaps from more familiar ones through liberal use of adhesives (cf. also Sibelius' Monument).

The chief choices are surely:

== To <u>choose</u> a parsing, or

== To list many (perhaps rank-ordered) in case of ambiguity.

If we select the first alternative, further choices are

- == to have a natural parsing (human).
- == to have a canonical parsing, in the sense of minimizing some variable (the minimization of the number of bodies leads us to Sibelius' Monument, its maximization to the Trivial Solution of the metatheorem [page 41]).

Other kinds of 2-dim data We have been discussing identification of 3-dim bodies (through their 2-dim projections) in a 2-dim scene, purely on the basis of geometric regions. Many other kinds of information could be used, such as texture, color, and shadows.

Nevertheless, it is interesting to see how far the identification of bodies can go if only geometric properties are used.

Conclusion Finding bodies in a 2-dim scene is a task <u>not very precisely defined</u>, because of the ambiguities inherent in any projection process: on these grounds, the concept of 'body' is best described through familiarity, human opinion and consensus. We are forced to this because any scene could be partitioned in several ways (cf. fig. 'PARALLELEPIPED') only some of which may be considered plausible or 'sensible' (natural, common, standard) partitions in regard to the bodies forming it.

TOTAL ANALYSIS OF VERTICES

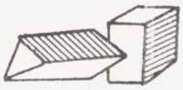
Synopsis Here a scene is considered as formed by several regions; bodies are adequate collections of regions. The problem of identifying bodies is re-stated as the problem of finding whether two regions belong or do not belong to the same body. This question is answered by examining the vertices of the scene.

It is shown that a single vertex never conveys conclusive evidence, so that at least a pair of vertices is required to isolate a body; familiar and unfamiliar configurations of objects help to understand how the vertices are to be used in this task.

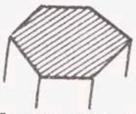
Vertices are the important feature

All faces on polyhedra are bounded by edges.
All edges terminate in vertices.

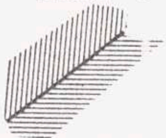
== This thesis deals with the analysis of visual scenes composed mainly by three-dimensional planar objects



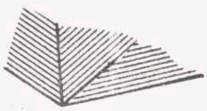
== These are limited by flat surfaces



== All these bodies share as a common feature the edge: place where two planes [faces] meet (but see page 57).



== Wherever several edges or faces meet, a vertex appears. This is also a common feature for all the bodies.



A body is formed by vertices with edges connecting some of these. When a 3-dim body is projected into a 2-dim body, its 3-dim vertices (which we will call genuine 3-dim vertices) are transformed into genuine 2-dim vertices, known as images of the 3-dim vertices, as figure 'GENUINE' (in next page) indicates.

That is, a genuine 2-dim vertex has come from a genuine 3-dim vertex. Some 2-dim "false" vertices appear too; they do not come

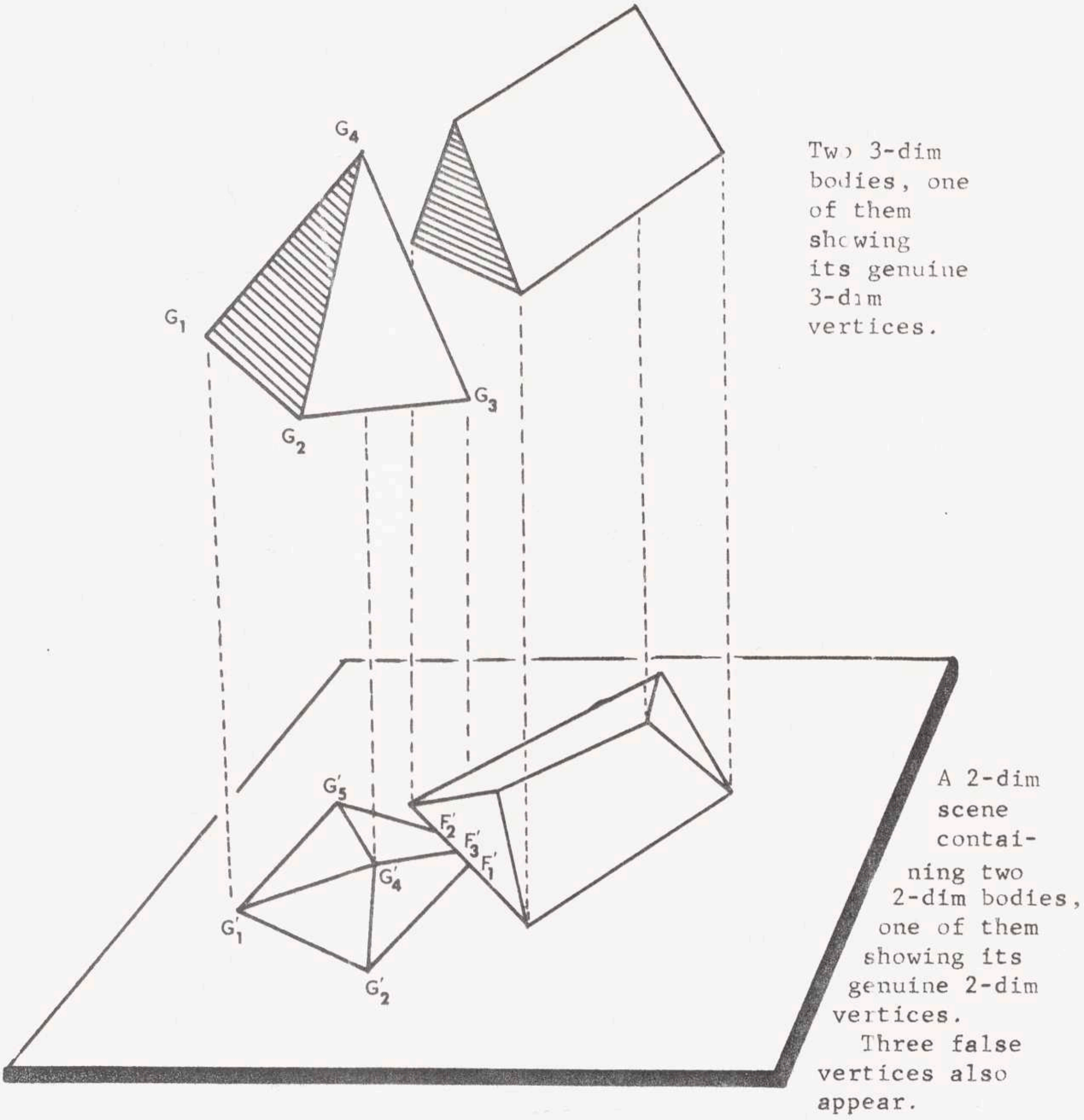


Figure 'G E N U I N E'

A genuine vertex (such as G_1') is one whose counterimage (G_1 in this case) belongs to some body; a false vertex such as F_2', is a virtual intersection, and generally has no counterimage in the 3-dim world. See fig. 'NODES'.

from genuine 3-dim vertices, but rather from the partial occlusion of parts of opaque bodies [transparent objects give rise to different kind of false vertices; Guzmán {MS Thesis} deals with them by using transparent models, and a mode of operation of TD, the recognizer, that re-interprets or ignores certain types of vertices. {AFCRL-67-0133}].

False vertices do not belong to any object.

<u>Genuine and false vertices</u> The classification of vertices into categories genuine and false will allow isolation of objects in a picture; in fig. 'GENUINE', elimination of vertices F_1', F_2', and F_3' divides the genuine nodes of the network (see fig. 'NODES') into two non-connected components, △ and □ , correctly separating the two bodies.

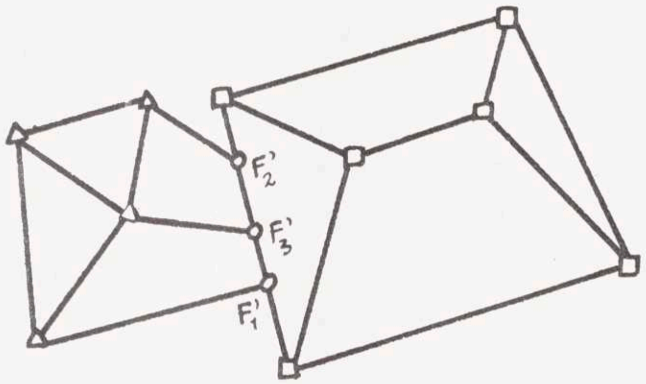


Figure 'N O D E S'
False vertices arise from the intersection of two projected edges, one of which is typically occluded in part by a face bordered by the other. Elimination of the false nodes F_1', F_2' and F_3' disconnects the network in two separate components, which are the bodies sought for.

This suggests the following

<u>2-dim body</u> (first approx. to definition). Set of regions possessing only genuine vertices, and separated from other bodies by false vertices.

In this way, the problem of identifying bodies is equivalent to the problem of identifying genuine vertices, segregating the false ones.

Problems to be solved The computation of this equivalence is challenged by several problems:

== The distribution and position of bodies may be such that false vertices look like genuine vertices (fig. 'CAUTION').

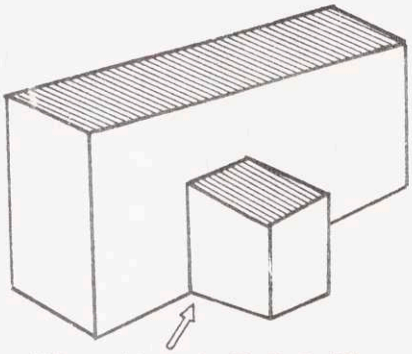
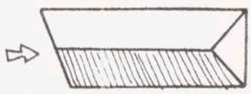


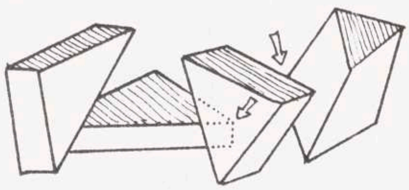
Fig. 'C A U T I O N'
That vertex looks genuine, but is false.

Global information (analysis of more than one vertex) is needed in general to distinguish them. In other words, although false vertices are those which separate two bodies, and 2-dim genuine vertices originate from 3-dim genuine vertices, to segregate them requires more than the simple analysis of their shape.

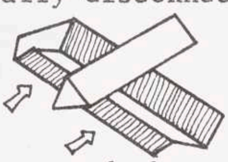
== Some genuine vertices look like false vertices.



== Genuine vertices of a body may not be present in the scene, or may be supplanted by false vertices.



== A single body may have totally disconnected sections (portions).



== Continuation is not clear: some doubts arise if the object in the foreground covers one or two bodies (fig. 'CONTINUATION'); the simplicity criterion prefers the single body interpretation.

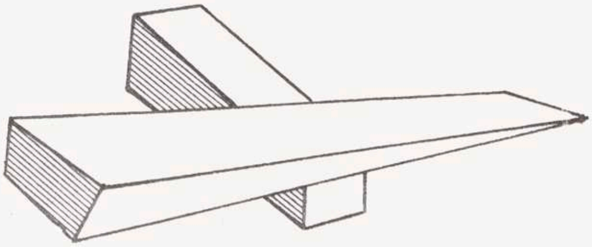


Fig. 'CONTINUATION'
Continuation is not clear.

In brief, difficulties are of two kinds:

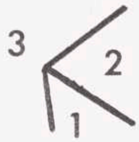
== Genuine and false vertices can not be distinguished locally (see Theorem below).

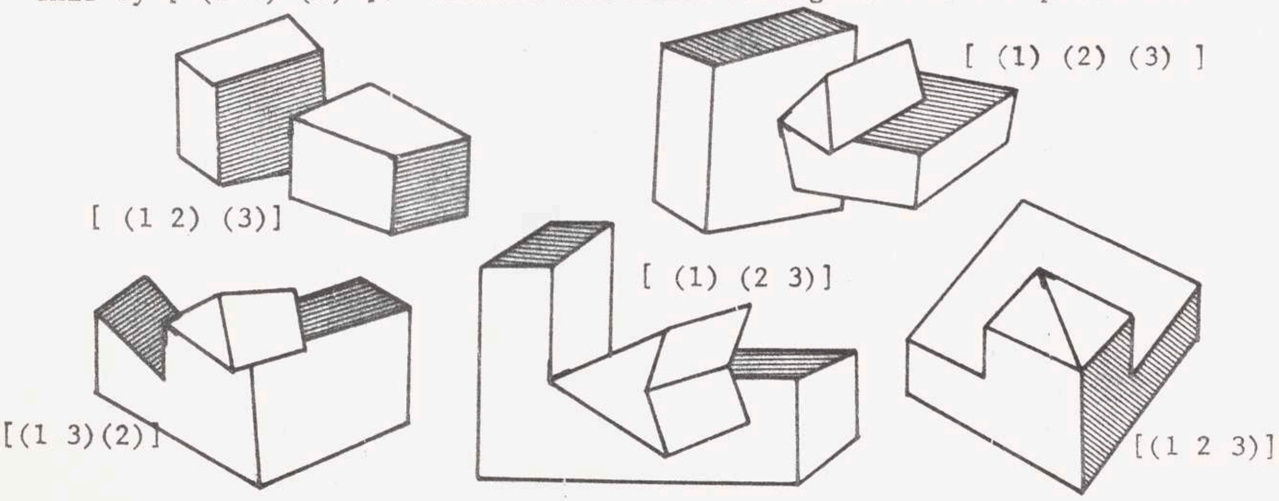
== Even when they are completely classified, problem of fig. 'CONTINUATION' remains.

The solution of these problems will have to make use of more global information.

Classification of Vertices The table 'VERTICES' in next page classifies vertices according to their form, number of lines and angles among the lines. It contains the most common types; vertices having more edges could have been included.

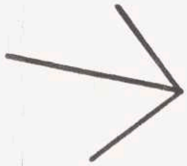
Let us consider one of these types, ARROW. Three regions called 1, 2, and 3, form it. The standard, most common ARROW configuration is a body with faces 1 and 2 seen against some other object 3. We indicate this by [(1 2) (3)]. However all other configurations are possible:





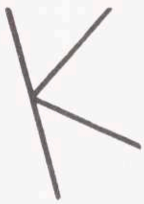
'L'.- Vertex where two lines meet.

'FORK'.- Three lines forming angles smaller than 180 °.

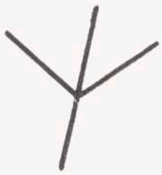


'ARROW'.- Three lines meeting at a point, with one of the angles bigger than 180 °.

'T'.- Three concurrent lines, two of them colinear.



'K'.- Two of the lines are colinear, 'th i and the other two fall in the same side of such lines.



'X'.- Two of the lines are colinear, and the other two fall on opposite sides of such lines.

'PEAK'.- Formed by four or more lines, when there is an angle bigger than 180°.

'MULTI'.- Vertices formed by four or more lines, and not falling in any of the preceding types.

TABLE 'V E R T I C E S'

Classification of rectilinear vertices.

Thus, for an ARROW, all the groupings of its faces are possible; any procedure that, by looking at an Arrow tries to decide how its faces are grouped into bodies, will always make mistakes.

The generalization of the above analysis to all other types of vertices proves the following

<u>"Theorem"</u>. There does not exist a set of local decision procedures $[\mu_i]$, each one looking or getting information from one vertex and establishing b-equivalences among some of their faces (two faces a and b are b-equivalent, indicated $a \equiv b$, if the μ_i decides that they belong to the same body; this is an equivalence relation), using information only from that vertex (it does not look at the other vertices or at the values of the μ's at the other vertices), which will partition all scenes correctly.

That is, the following machine will not work for all scenes:

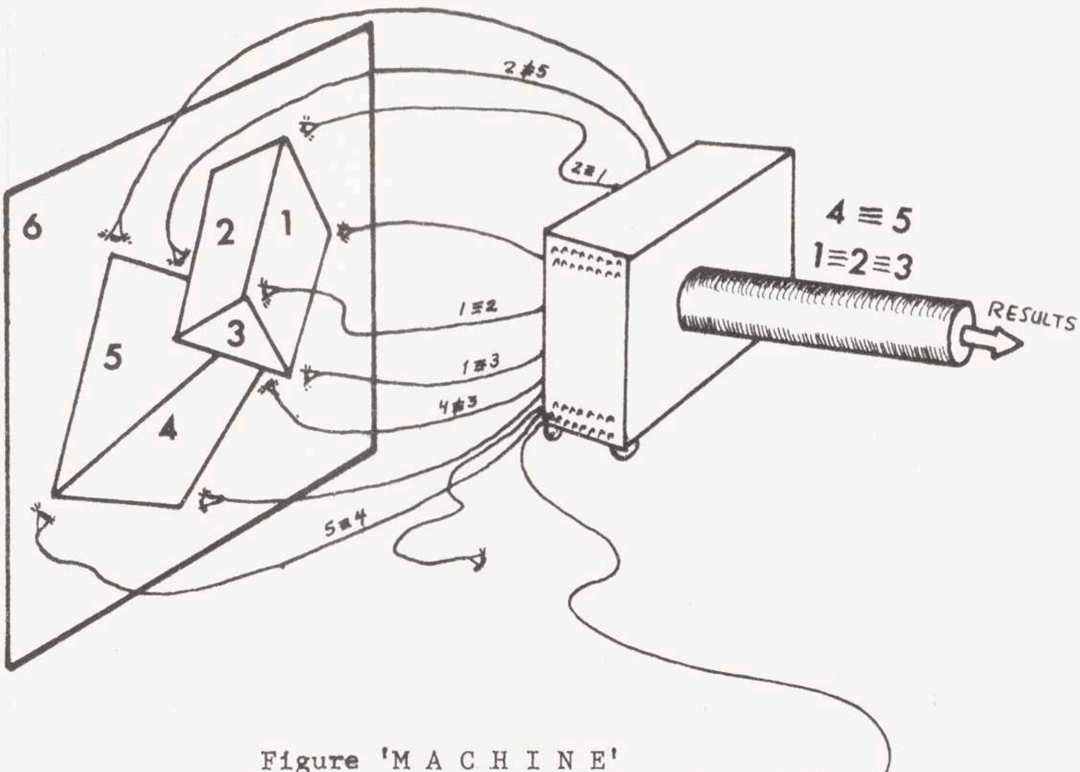


Figure 'M A C H I N E'

The decision procedures μ_i , represented as 'eyes' here, decide by processing information at exactly one vertex; the box in the left accepts all these decisions and passes them as results. No matter what set of μ_i we choose, there exists a scene that induces an incorrect partition by our machine.

A stronger assertion is that, in view of inherent ambiguity, there is not even any global procedure!

All the different groupings of regions of a vertex into bodies are possible; this is illustrated by the following complete set of scenes, each one of them showing a different partitioning of a type of vertex. These examples are useful also in giving an idea of unusual, as well as familiar scenes; we will have later occasion to use them, when searching for heuristics to form bodies.

Generation of partitions

There are only two partitions of a set of two elements. 2

```
    compo (   (1 2)   )
1     ((1) (2))
2     ((1 2))
      2
```

Partitions of a set of 4 elements

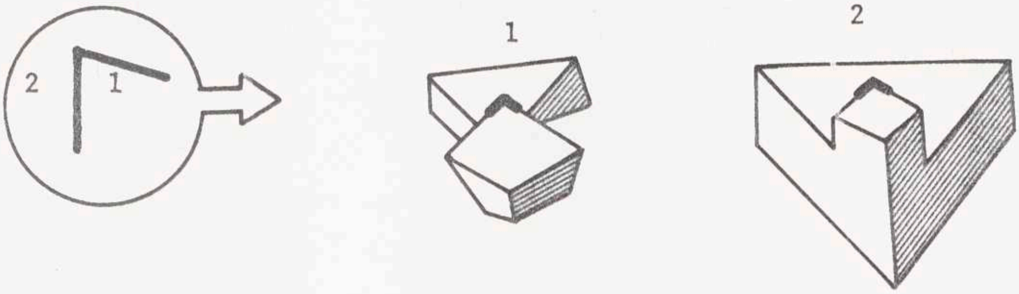
```
    compo (   (1 2 3 4)   )
1     ((1) (2) (3) (4))
2     ((1 2) (3) (4))
3     ((1 3) (2) (4))
4     ((1 4) (2) (3))
5     ((1) (2 3) (4))
6     ((1 2 3) (4))
7     ((1 4) (2 3))
8     ((1) (2 4) (3))
9     ((1 2 4) (3))
10    ((1 3) (2 4))
11    ((1) (2) (3 4))
12    ((1 2) (3 4))
13    ((1 3 4) (2))
14    ((1) (2 3 4))
15    ((1 2 3 4))
      15
```

Figures in the next few pages are numbered according to the numbers in the leftmost column in these tables.

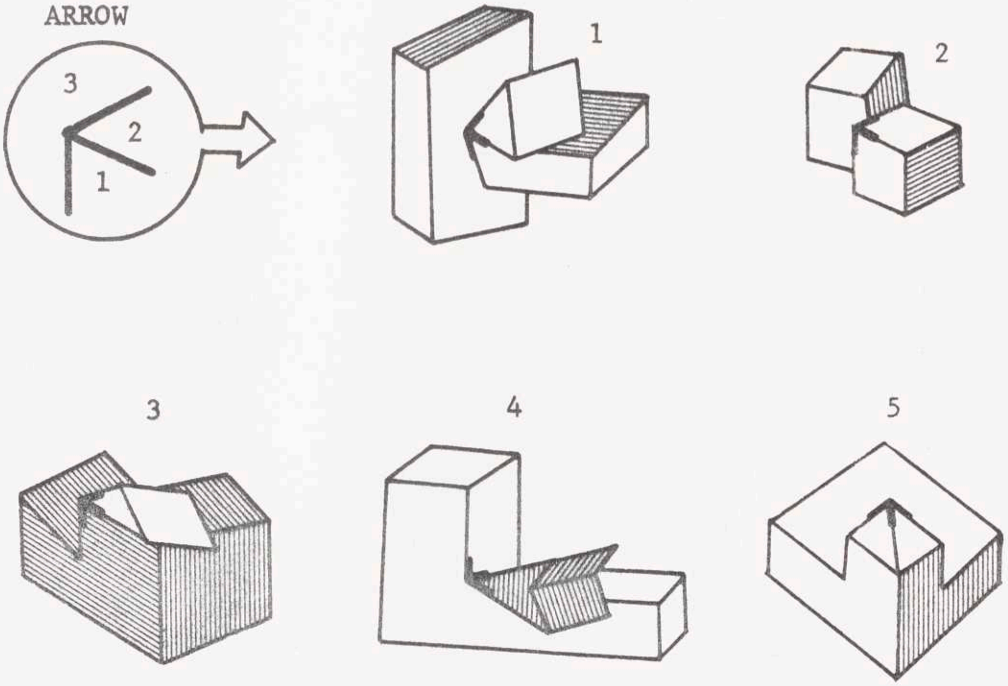
Partitions of a set of 3 elements

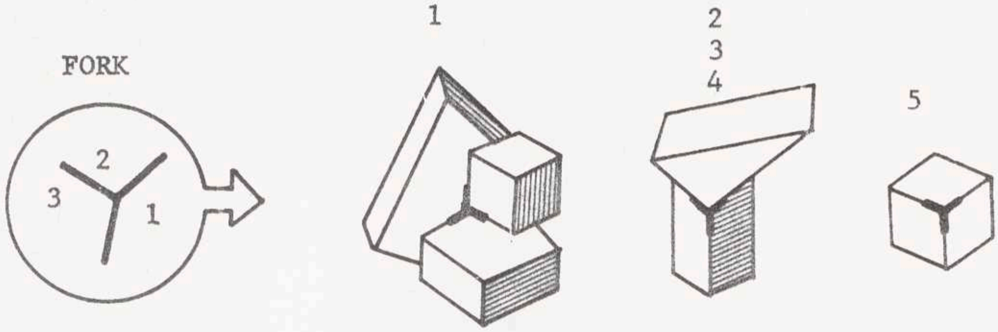
```
    compo (   (1 2 3)   )
1     ((1) (2) (3))
2     ((1 2) (3))
3     ((1 3) (2))
4     ((1) (2 3))
5     ((1 2 3))
      5
```

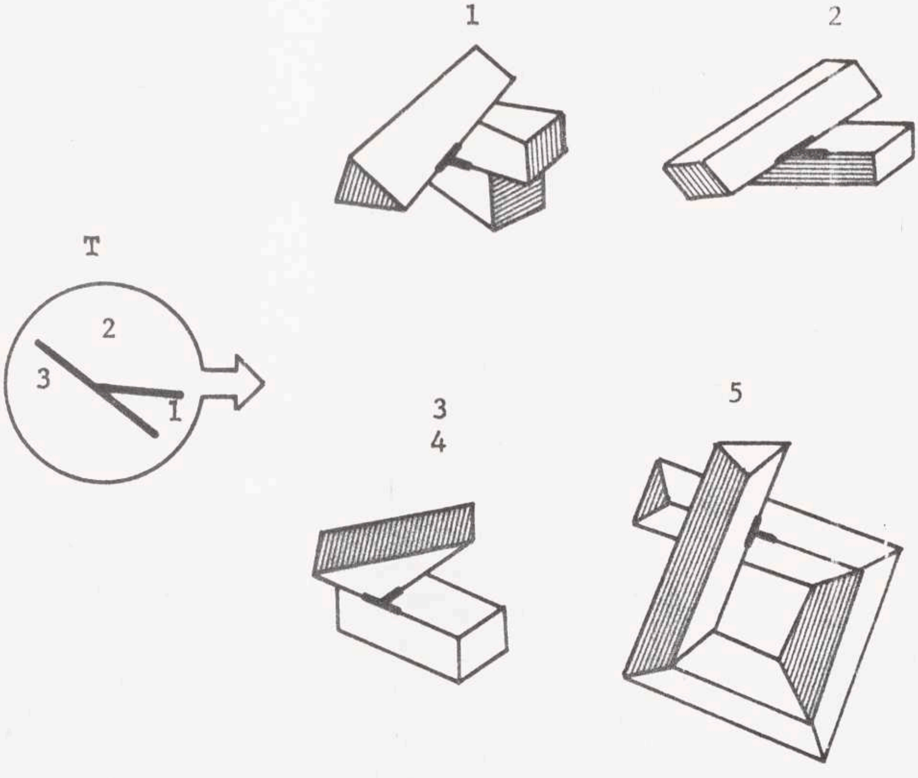
CORNER

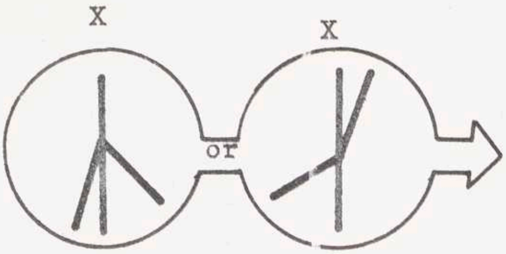


ARROW

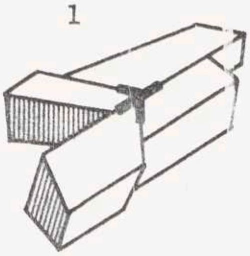



FORK
1
2
3
1
2
3
4
5

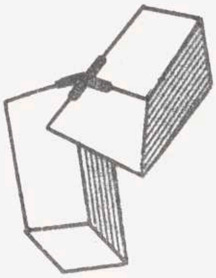

T
1
2
2
3
1
3
4
5



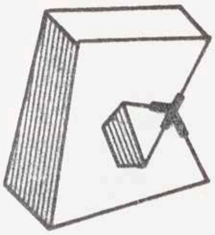
1



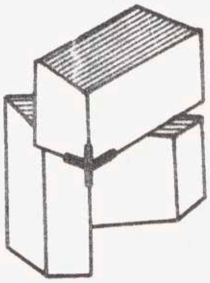
2
11



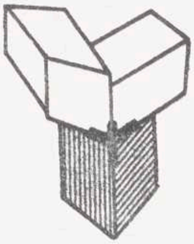
3
8



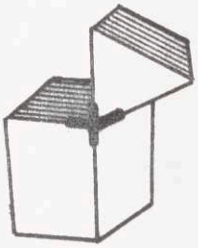
4



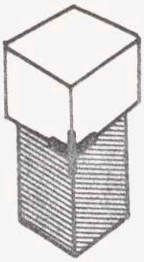
5



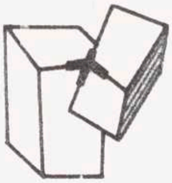
6
14



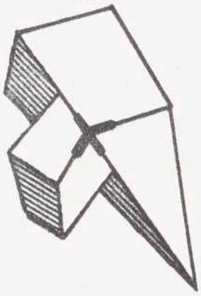
7



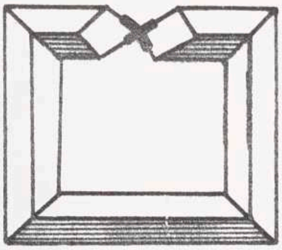
12



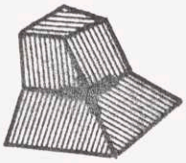
9
13

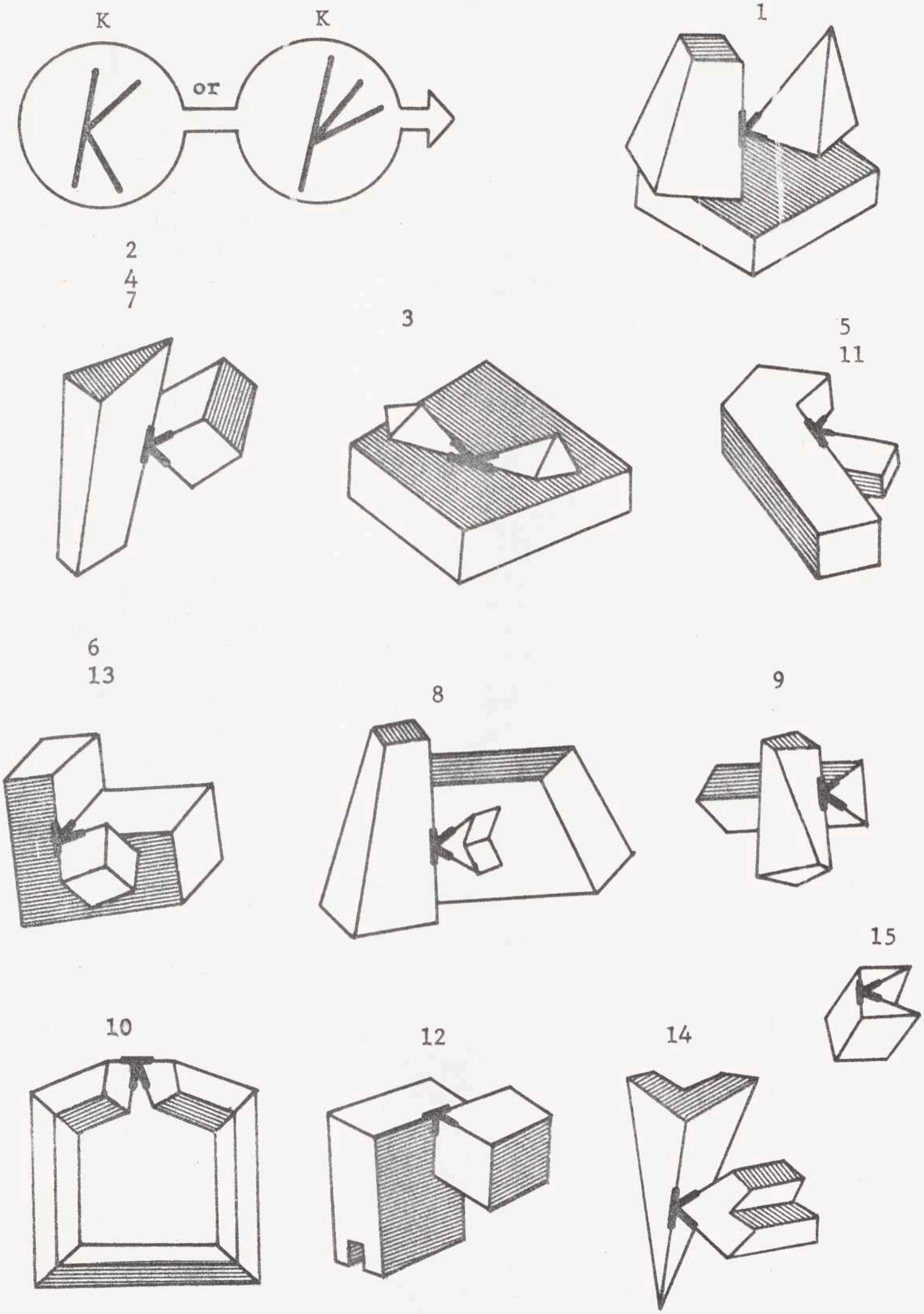


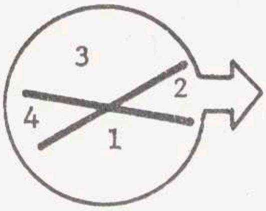
10



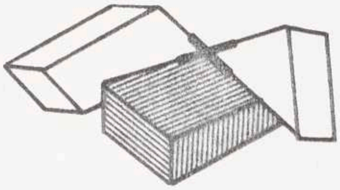
15



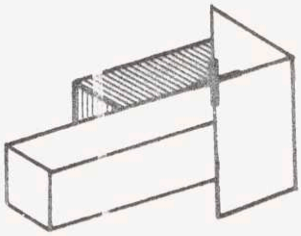

K
K
or
1
2
4
7
3
5
11
6
13
8
9
15
10
12
14



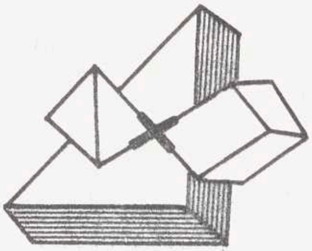
1



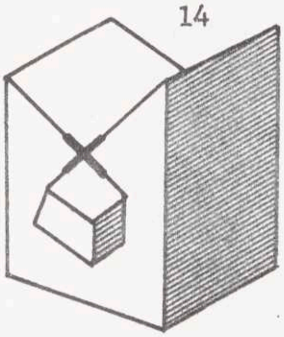
2
4
5
11



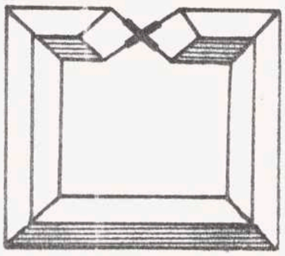
3
8



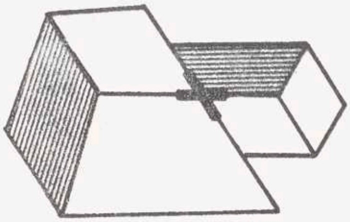
6
9
13
14



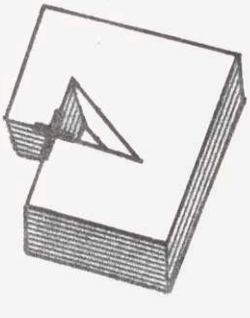
10



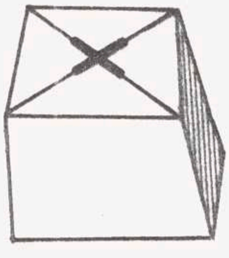
7
12



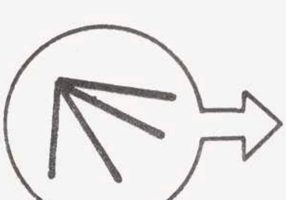
15



15

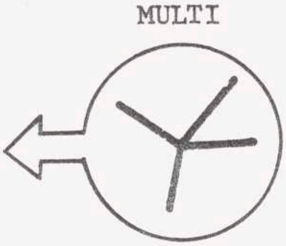


PEAK



not
represen
ted
h e r e

MULTI



Digression 1. An alternate approach

As an alternate approach, one could try to use the faces as a basis for identification. For instance, use two scenes (left image, right image) or pictures, localize a sharp feature in one of them (vertex, crack in the face, peculiar texture, etc.) and by correlation or some other method, find it also in the other picture. Having found a few points in both images in this manner, determine the plane of the face, in 3-dim space. When several faces are thus identified, we can compute, if desired, their intersection and obtain the edges (lines). It will generally suffice to ignore the edges and rely on the faces. Since it is reasonable to expect considerable difficulty in finding lines and in differentiating lines caused by edges from those caused by shadows, an apprach which avoids the lines altogether looks promising. But in this case, in addition to requiring two images, several correlations are needed (if we choose this method), a generally time-consuming and error-prone task.

S E E, A PROGRAM THAT FINDS BODIES IN A SCENE

Synopsis

It is explained how SEE works.

Algorithms and heuristics are presented, implemented in a program, that analyze a scene into a composition of three-dimensional objects. Only the two-dimensional representation of the three-dimensional scene is available as input, and is described by a collection of surfaces, lines and vertices.

SEE looks for three-dimensional objects in two-dimensional scenes. The program does not require a pre-conceived idea of the form of the objects which could appear in the scenes. It is only assumed that they will be solid objects formed by plane surfaces. Thus, SEE can not find "pentagonal prisms" or "houses" in a scene, since it does not know what a "pentagonal prism" is; but **it will usually isolate the pentagonal prisms** (or any other regular or irregular solid) **in a scene, even if some of them are partially occluded, without having a description of such objects.** It does this by paying attention to configuration of surfaces and lines which would make plausible three-dimensional solids, and in this way 'bodies' are identified.

The analysis that SEE makes of the different scenes generally agrees with human opinion, although in some ambiguous cases they tend to be conservative. The most interesting thing about the program is how well it deals with occlusions. Many examples in the next section 'Analysis of many scenes' illustrate the features and peculiarities of the program, and also illustrate the effects of inaccuracies introduced in the data.

L 1 0

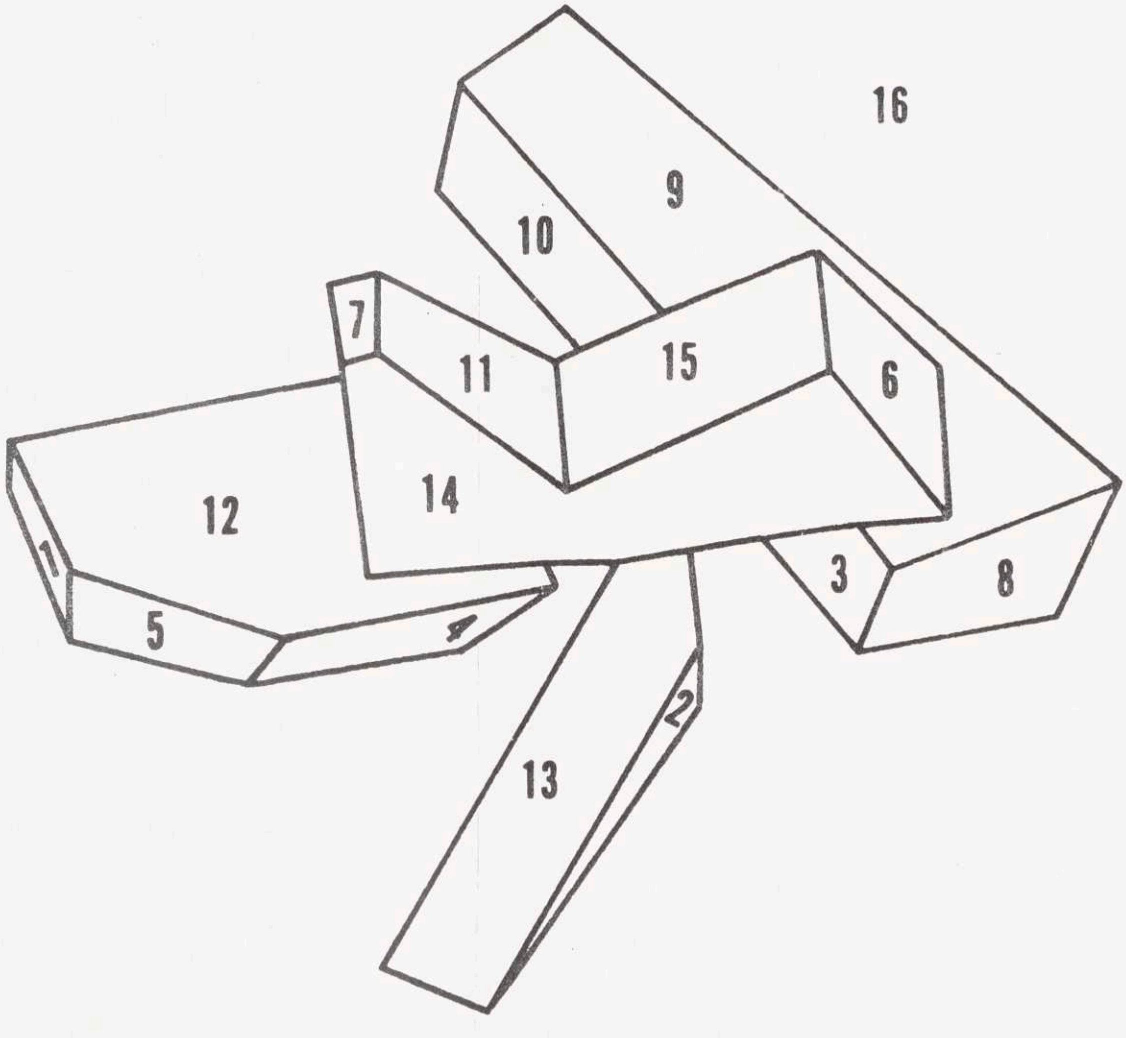


FIGURE 'L 1 0'

A scene analyzed by SEE.

INTRODUCTION

Here is a program that locates objects in an optical image of a scene most likely composed by three-dimensional solids, perhaps occluding one to another, so that some of them may not be totally visible. We use a line drawing as our representation of the scene.

The analysis of scene L10 (see figure 'L10' in next page) by our program, named SEE, produces

```
(BODY 1. IS :5 :1 :4 :12)
(BODY 2. IS :6 :15 :7 :11 :14)
(BODY 3. IS :8 :9 :10 :3)
(BODY 4. IS :2 :13)
```

Division of work in computer vision

In trying to construct a program for seeing, several approaches are possible; most of them require some of the following set of modular programs or subroutines.

Pre-processing. Converts the image from a 2-dim array of intensities to a symbolic representation or 'internal format' (page 66), in terms of vertices and lines connecting them.

Homogeneity predicates. They decide if areas of the picture are inhomogeneous, and hence require further analysis (page 16).

Color predicates. Boundaries of different color suggest lines.

Line finder. Locates lines of points having certain property (such as being inhomogeneous, or having a large light intensity gradient).

Vertex finder. Concurrent lines are merged, or a vertex is created at their meeting point.

Consolidator. Eliminates the false lines and finds more lines, incrementing in this way as much as possible the reliability of the system.

Illumination program. Discovers where the main light sources are.

Shadows program. Detects shadows so as to eliminate them.

Missing lines program. General shape considerations suggest places where faint lines can remain undetected.

Body recognition. Partitions the scene into appropiate subsets, each one being a body or object. Thus, SEE is a body-recognition program.

Object identification. These objects are compared against abstract descriptions (models) of cubes, pyramids, etc., so that a classification is done, and a name is attached to each one. In the process, certain parameters may acquire values: the height of the pyramid is observed.

Positioning. Having analyzed the scene, the relevant objects are positioned in three-dimensional space, and additional relations among them are discovered (support, obstruction, etc.). Enough information is obtained to allow the mechanical arm to manipulate the objects and achieve its goals.

Stereo. More than one view are analyzed (page 233) and from them, 3-dim spatial positions are found.

Focussing. The computer, by adjusting the focus of its lens, acquires knowledge of how far the objects are.

Feedback among these parts is more necessary as the complexity of the scene and of the desired goals increases.

Recognizer. The task of body recognition and body identification was formerly accomplished by a single program (for instance, DT or TD {my MS Thesis}) that compares the symbolic description of the scene against the symbolic or abstract description of the model of the desired object, in a kind of two-dimensional matching, to isolate instances of that object in the scene.

Technical descriptions of SEE

1. Annotated listings. Above all, the primary source of information is the listing of the programs, that appears complete in this thesis. They are written in Lisp. If, despite my efforts, some of my explanations are not clear, consult it: it is annotated. The programs themselves,

examples, test data, results, instructions, etc., are in the DEC-magnetic tape "GUZMAN F" at Project MAC (AI group). Instructions are here in page 78.

2. This section of the thesis contains a description and discussion of the different algorithms and procedures used.

3. Published papers that cover part of the material at somewhat less depth, and therefore are more readable, are also available {FJCC 68} {Pisa 68}. Except that they contain some examples not included here, they contain no other information not covered here.

4. An internal report {MAC M 357} described an earlier version of SEE.

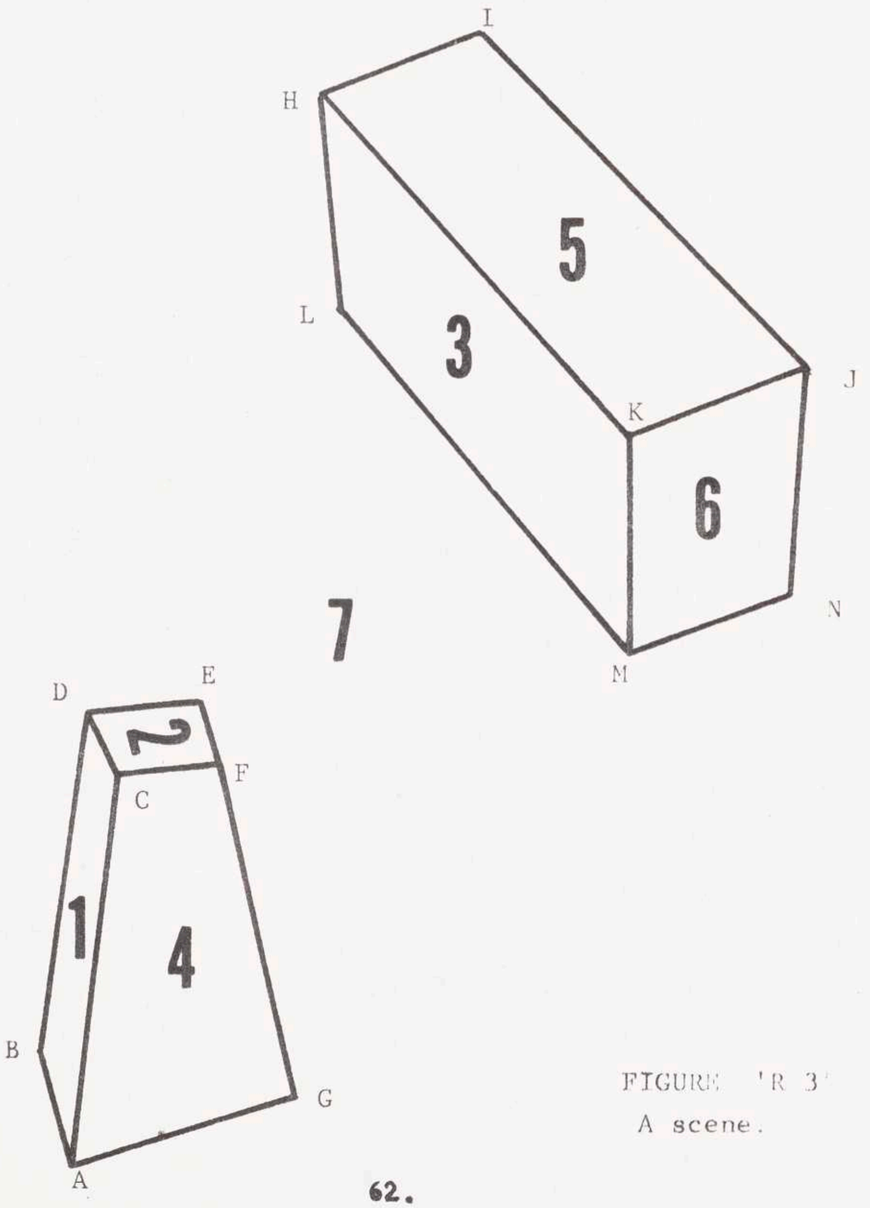


FIGURE 'R 3'
A scene.

I N P U T   F O R M A T

Eventually, several preprocessors will be able to receive data through an input camera and reduce it to the "internal format" of a scene, in the form required by SEE. For testing purposes, the scenes are entered by hand in a simplified format, called 'input format', to be described now. <u>All the scenes analyzed by SEE have been written in input format</u>.

Example. R3 . The input format of scene R3 is

```
(DEFPROP R3 (%:7) BACKGROUND)
(NOT (SETQ R3 (QUOTE (
%A   4.3    4.5    (%:7 %G %:4 %C %:1 %B)
%B   4.0    5.7    (%:7 %A %:1 %D)
%C   4.8    8.5    (%:4 %F %:2 %D %:1 %A)
%D   4.5    9.15   (%:7 %B %:1 %C %:2 %E)
%E   5.65   9.25   (%:7 %D %:2 %F)
%F   5.85   8.6   (%:7 %E %:2 %C %:4 %G)
%G   6.6    5.2   (%:7 %F %:4 %A)
%H   6.9   15.4   (%:7 %L %:3 %K %:5 %I)
%I   8.5   16.0   (%:7 %H %:5 %J)
%J  11.8   12.6   (%:7 %I %:5 %K %:6 %N)
%K  10.0   11.9   (%:6 %J %:5 %H %:3 %M)
%L   7.1   13.2   (%:7 %M %:3 %H)
%M  10.0    9.7   (%:7 %N %:6 %K %:3 %L)
%N  11.65  10.3   (%:7 %J %:6 %M)
))))
```

R3 IN INPUT FORMAT

The first line declares :7 to be the background. We have to tell SEE which regions belong to the background. If this information is missing, a program is called that will compute the regions that belong to the background (see section 'Background discrimination by computer') prior to other calculations.

After that, the lines associate with each vertex its 2-dim coordinates and a list (which will later be called 'KIND'), in counterclockwise order, of regions and vertices radiating from that vertex.

The function PREPARA (see listing) converts the scene as just given to the "internal format" form which SEE expects. It does this by putting many properties in the property lists of the atoms representing vertices and regions (property lists in Lisp get explained in page 65).

Property lists in Lisp * Each atomic expression in Lisp has a property list, which is a place where facts can be stored.

If it is desired to represent the fact that John is a 69 years old male, has a wife called Jacqueline, and a height of value 1.77 m, we could proceed in Lisp as follows:

(1) We will agree that the atom 'JOHN' will represent our man.

(2) In the property list of 'JOHN' we will store several properties or indicators and their values, using the function PUTPROP, that stores information in the property list; thus
(Putprop (quote John) (quote Jacqueline) (quote Wife))
will add, under the indicator or property 'Wife', the value 'Jacqueline':

```
JOHN
  |
  WIFE ———————— JACQUELINE
```

(3) Hence, the representation of our facts in Lisp is

```
JOHN
  |
  SEX      --   MALE
  |
  AGE      --   69.0
  |
  WIFE     --   JACQUELINE
  |
  HEIGHT   --   (1.77 m)
```

(4) In fact, the property list of 'JOHN', which is the CDR of 'JOHN' in Lisp 1.6 {MAC M 313}, is

(SEX MALE   AGE 69.0   WIFE JACQUELINE   HEIGHT (1.77 m) ...)

(5) If later we want to know the age of John, we will ask

(Get (quote John) (quote Age))

and the value will be 69.0

---

* This paragraph, which can be skipped if it is known what a property list is, will make the next section clearer.

```
FORMAT OF SCENE  R3
R3
      REGIONS          (%:6 %:5 %:3 %:2 %:1 %:4 %:7)
      VERTICES         (%N %M %L %K %J %I %H %G %F %E %D %C %B
 %A)
      BACKGROUND       (%:7)
%:6
      NEIGHBORS     (%:5 %:3 %:7 %:7)
      KVERTICES     (%K %M %N %J)
      FOOP          ((%:5 %K %:3 %M %:7 %N %:7 %J))
%:5
      NEIGHBORS     (%:3 %:6 %:7 %:7)
      KVERTICES     (%K %J %I %H)
      FOOP          ((%:3 %K %:6 %J %:7 %I %:7 %H))
%:3
      NEIGHBORS     (%:7 %:7 %:6 %:5)
      KVERTICES     (%L %M %K %H)
      FOOP          ((%:7 %L %:7 %M %:6 %K %:5 %H))
%:2
      NEIGHBORS     (%:4 %:7 %:7 %:1)
      KVERTICES     (%F %E %D %C)
      FOOP          ((%:4 %F %:7 %E %:7 %D %:1 %C))
%:1
      NEIGHBORS     (%:4 %:2 %:7 %:7)
      KVERTICES     (%C %D %B %A)
      FOOP          ((%:4 %C %:2 %D %:7 %B %:7 %A))
%:4
      NEIGHBORS     (%:2 %:1 %:7 %:7)
      KVERTICES     (%C %A %G %F)
      FOOP          ((%:2 %C %:1 %A %:7 %G %:7 %F))
%:7
      NEIGHBORS     (%:6 %:6 %:3 %:3 %:5 %:5 %:2 %:2 %:4 %:4
%:1 %:1)
      KVERTICES     (%N %M %L %H %I %J %E %F %G %A %B %D)
      FOOP          ((%:6 %N %:6 %M %:3 %L %:3 %H %:5 %I %:5
%J %E %F %G %A %B %D) (%:2 %E %:2 %F %:4 %G %:4 %A %:1 %B %:
1 %D))
%N
       XCOR        11.649999
       YCOR        10.299999
       NVERTICES   (%J %M)
       NREGIONS    (%:7 %:6)
       KIND        (%:7 %J %:6 %M)
       TYPE        (L (%:6 %:7))
%M
       XCOR        10.0
       YCOR        9.7000000
       NVERTICES   (%N %K %L)
       NREGIONS    (%:7 %:6 %:3)
       KIND        (%:7 %N %:6 %K %:3 %L)
       TYPE        (ARROW (%K %M %N %L %:6 %:3 %:7))
%L
       XCOR        7.1000000
       YCOR        13.200000
       NVERTICES   (%M %H)
       NREGIONS    (%:7 %:3)
       KIND        (%:7 %M %:3 %H)
       TYPE        (L (%:3 %:7))
%K
       XCOR        10.0
       YCOR        11.899999
       NVERTICES   (%J %H %M)
       NREGIONS    (%:6 %:5 %:3)
       KIND        (%:6 %J %:5 %H %:3 %M)
       TYPE        (FORK %K)
%J
       XCOR        11.799999
       YCOR        12.600000
       NVERTICES   (%I %K %N)
       NREGIONS    (%:7 %:5 %:6)
       KIND        (%:7 %I %:5 %K %:6 %N)
       TYPE        (ARROW (%K %J %I %N %:5 %:6 %:7))
%I
       XCOR        8.5
       YCOR        16.0
       NVERTICES   (%H %J)
       NREGIONS    (%:7 %:5)
       KIND        (%:7 %H %:5 %J)
       TYPE        (L (%:5 %:7))
%H
       XCOR        6.8999999
       YCOR        15.399999
       NVERTICES   (%L %K %I)
       NREGIONS    (%:7 %:3 %:5)
       KIND        (%:7 %L %:3 %K %:5 %I)
       TYPE        (ARROW (%K %H %L %I %:3 %:5 %:7))
%G
       XCOR        6.6000000
       YCOR        5.1999999
       NVERTICES   (%F %A)
       NREGIONS    (%:7 %:4)
       KIND        (%:7 %F %:4 %A)
       TYPE        (L (%:4 %:7))
%F
       XCOR        5.8500000
       YCOR        8.6000000
       NVERTICES   (%E %C %G)
       NREGIONS    (%:7 %:2 %:4)
       KIND        (%:7 %E %:2 %C %:4 %G)
       TYPE        (T (%C %F %G %E %:2 %:4 %:7))
%E
       XCOR        5.6499999
       YCOR        9.25
       NVERTICES   (%D %F)
       NREGIONS    (%:7 %:2)
       KIND        (%:7 %D %:2 %F)
       TYPE        (L (%:2 %:7))
%D
       XCOR        4.5
       YCOR        9.1499999
       NVERTICES   (%B %C %E)
       NREGIONS    (%:7 %:1 %:2)
       KIND        (%:7 %B %:1 %C %:2 %E)
       TYPE        (ARROW (%C %D %B %E %:1 %:2 %:7))
%C
       XCOR        4.8000000
       YCOR        8.5
       NVERTICES   (%F %D %A)
       NREGIONS    (%:4 %:2 %:1)
       KIND        (%:4 %F %:2 %D %:1 %A)
       TYPE        (FORK %C)
%B
       XCOR        4.0
       YCOR        5.6999999
       NVERTICES   (%A %D)
       NREGIONS    (%:7 %:1)
       KIND        (%:7 %A %:1 %D)
       TYPE        (L (%:1 %:7))
%A
       XCOR        4.3000000
       YCOR        4.5
       NVERTICES   (%G %C %B)
       NREGIONS    (%:7 %:4 %:1)
       KIND        (%:7 %G %:4 %C %:1 %B)
       TYPE        (ARROW (%C %A %G %B %:4 %:1 %:7))
```

TABLE

R 3   IN   INTERNAL   FORMAT

I N T E R N A L   F O R M A T

The program assumes the scene in a special symbolic format, which basically, is an arrangement of relations between vertices and regions, which are represented by atoms having adequate properties in their property-lists. (See 'property lists in Lisp', page 64).

A scene has a name which identifies it; this name is an atom whose property lists contains the properties 'REGIONS', 'VERTICES', and 'BACKGROUND'. For example, the scene R3 (see figure R3) has the name 'R3'. In the property list of R3 we find (see also table "R3 IN INTERNAL FORMAT")

```
REGIONS        (%:6 %:5 %:3 %:2 %:1 %:4 %:7)
```

Unordered list of regions composing the scene R3. Order is immaterial.

```
VERTICES       (%N %M %L %K %J %I %H %G %F %E %D %C %B %A)
```

Unordered list of vertices composing the scene R3.

```
BACKGROUND     (%:7)
```

Unordered list of regions composing the background of scene R3.

Region A region corresponds to a surface limited by simple closed curves. Regions are represented by atoms that start with a colon (:). For instance, in R3, the surface delimited by the vertices K J N M is a region, called :6, but D E F G A C is not.

Each region has as name an atom which possess additional properties describing different attibutes of the region in question. These are 'NEIGHBORS', 'KVERTICES', and 'FOOP'. For example, the region in scene R3 formed by the lines DE, EF, FC, CD has ':2' as its name. In the property list of :2 we find:

```
NEIGHBORS      (%:4 %:7 %:7 %:1)
```

Counterclockwise ordered list of all regions which are neighbors to :2. For each region, this list is unique up to cyclic permutation.

```
KVERTICES     (%F %E %D %C)
```

Counterclockwise ordered list of all vertices which belong to region :2. This list is unique up to cyclic permutation.

```
FOOP          ((%:4 %F %:7 %E %:7 %D %:1 %C))
```

Each sublist is a counterclockwise ordered list of alternating neighbors and kvertices of :2. Each sublist is unique up to cyclic permutation, and indicates a simple boundary.

Each sublist of 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.

As other example, in the property list of :7 we find:

```
    NEIGHBORS     (%:6 %:6 %:3 %:3 %:5 %:5 %:2 %:2 %:4 %:4
%:1 %:1)
    KVERTICES     (%N %M %L %H %I %J %E %F %G %A %B %D)
    FOOP          ((%:6 %N %:6 %M %:3 %L %:3 %H %:5 %I %:5
%J %E %F %G %A %B %D) (%:2 %E %:2 %F %:4 %G %:4 %A %:1 %B %:
1 %D))
```

__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 R3. Each vertex has as name an atom which possess additional properties describing different attributes of the vertex in question. These are 'XCOR', 'YCOR', 'NVERTICES', 'NREGIONS', 'KIND', 'TYPE', and 'NEXTE'. For example, vertex J (see scene R3) has in its property list:

```
XCOR       11.799999
```

x-coordinate

```
YCOR       12.600000
```

y-coordinate

```
NVERTICES  (%I %K %N)
```

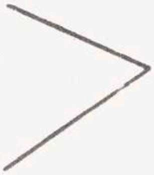
Counterclockwise ordered list of vertices to which J is connected. Unique up to cyclic permutation.

NREGIONS	(%:7 %:5 %:6)	Counterclockwise ordered list of regions to which J is connected. Unique up to cyclic permutation.
KIND	(%:7 %I %:5 %K %:6 %N)	Counterclockwise ordered list of alternating nregions and nvertices of J. This list is unique up to cyclic permutation.
TYPE	(ARROW (%K %J %I %N %:5 %:6 %:7))	List of two elements; the first is an atom indicating the <u>type-name</u> of J; the second is the <u>datum</u> of J. To be explained in next section.
(NEXTE)		Vertex J does not have the indicator NEXTE in its property list.

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, by taking its odd positioned elements, and its even positioned elements, respectively.

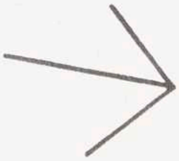
NEXTE is a property that appears in certain vertices (none in scene R3); it will be explained in next section.

The property TYPE is also put by the function PREPARA; it classifies each vertex into one of several types, as described below.



'L'.- Vertex where two lines meet.

'FORK'.- Three lines forming angles smaller than 180 °.



'ARROW'.- Three lines meeting at a point, with one of the angles bigger than 180 °.

'T'.- Three concurrent lines, two of them colinear.

'K'.- Two of the *lines* are colinear, and the other two fall on the same side of such lines.

'X'.- Two of the *lines* are colinear, and the other two fall on opposite sides of such lines.

'PEAK'.- Formed by four or more lines, when there is an angle bigger than 180°.

'MULTI'.- Vertices formed by four or more lines, and not falling in any of the preceding types.

TABLE 'V E R T I C E S'

Classification of rectilinear vertices.

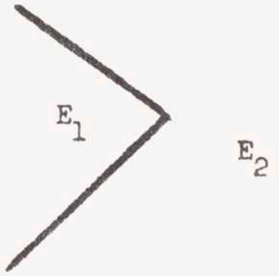
T Y P E S   O F   V E R T I C E S

The disposition, slope and number of lines which form a vertex are used to classified it, task performed by the function (TYPEGENERATOR L) by storing in its property list its corresponding type.

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

Vertices where two lines meet.

L.- A vertex formed by only two lines is always classified as of type 'L'. Two angles exist at it, one bigger and other smaller than 180°. The datum is a list of the form



(E_1 E_2), where E_1 is the region which contains the angle smaller than 180°.

E_2 is the region which contains the angle greater than 180°.

For instance, in scene R3 (see fig. ' R3 '), G has in its property list:

```
TYPE          (L (%:4 %:7))
```

The vertices of type L present in R3 are B, E, G, I, L, N.

Vertices where three lines meet.

FORK.- Three lines meeting at a point and forming angles smaller than 180° form a FORK.

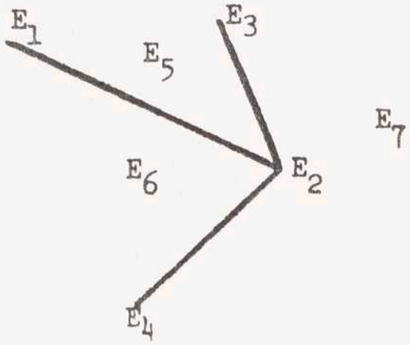
Its datum is the vertex itself at which the fork occurs. For instance, vertex K has in its property list

```
TYPE          (FORK %K)
```

The vertices of type FORK present in R3 are C, K.

ARROW.- Three lines meeting at a point, with one of the angles bigger than 180°.

The datum of an ARROW is a list like (E_1 E_2 E_3 E_4 E_5 E_6 E_7) where



E_1 is the vertex at the 'tail'.

E_2 is the vertex at the center

E_3 is the vertex at the left of $E_1 \to E_2$.

E_4 is the vertex at the right.

E_5 is the region at the left.

E_6 is the region at the right

E_7 is the region which contains the angle bigger than 180^0.

For instance, vertex H has in its property list

```
TYPE          (ARROW (%K %H %L %I %:3 %:5 %:7))    --fig. R3 --
```

The vertices of type ARROW present in R3 are A, D, H, J, M.

T.- Three concurrent lines, of which two are colinear.

The datum for a T is a list of the form (E_1 E_2 E_3 E_4 E_5 E_6 E_7), where

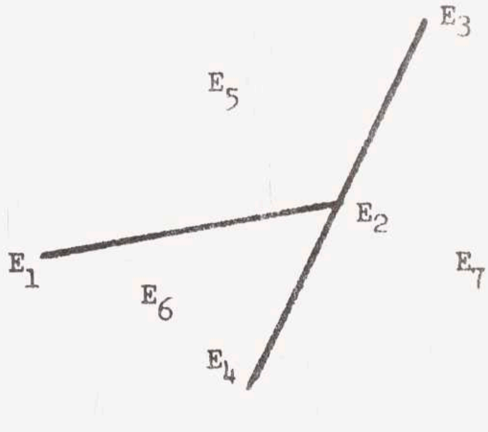
E_1 is the vertex at the 'tail' of the T.

E_2 is the central vertex.

E_3 is a vertex such that E_1 E_2 E_3 is an angle between 90 and 180 degrees.

E_4 is a vertex such that E_1 E_2 E_4 is an angle smaller than 90 degrees. That is, E_3 E_4 E_2 are colinear.

E_5 is the region which contains the angle between 90 and 180 degrees.



E_6 is the region which contains the angle smaller than 90 degrees.

E_7 is the "central" region (where the 180° angle is).

For instance, vertex F (fig. R3) has in its property list

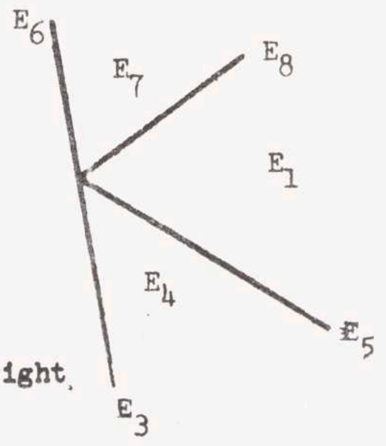
```
TYPE          (T (%C %F %G %E %:2 %:4 %:7))
```

The vertices of type T present in R3 are F only.

See also "Matching T's or Nextes" below.

<u>Vertices where four lines meet.</u>

K.- When two of the *lines* are colinear, and the other two fall in the same side of such lines. The datum is a list of the form (E_1 E_2 E_3 E_4 E_5 E_6 E_7 E_8) where



E_1 is the central region.

E_2 is the region having the 180° angle.

E_3 is the colinear vertex which falls to the left of E_1 E_2.

E_4 is the region to the left of $E_1 \rightarrow E_2$

E_5 is the vertex to the left of $E_1 \rightarrow E_2$

E_6 is the colinear vertex which falls to the right.

E_7 is the region to **the right.** of $E1 \rightarrow E_2$.

E_8 is the other vertex to the right (of E_1).

R3 contains no vertices of type K. PA of figure BRIDGE is of type 'K'.

X.- When two of the *lines* **are colinear,** and the other two **fall in opposite sides of such** lines. **The datum is a list of the form (E_1 E_2 E_3 E_4 E_5 E_6), where**

E_1 is one of the colinear vertices.

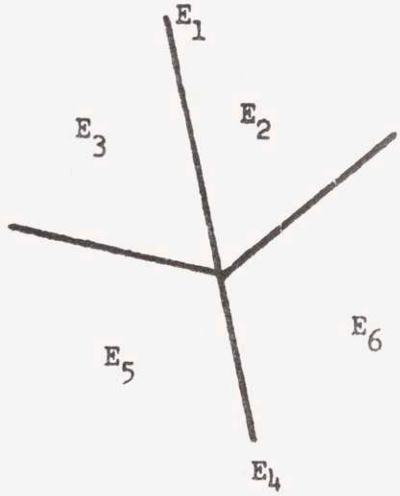
E_2 is the region to the left of E_1 C,
where C is the vertex at the center.

E_3 is the region to the right of E_1 C,

E_4 is the other colinear vertex.

E_5 is the region to the left of E_4 C.

E_6 is the region to the right of E_4 C.



For instance, we find in the property list of F (figure BRIDGE):

TYPE (X (QA :26 :22 G :21 :30))

The vertices of type X present in BRIDGE **are** F, only.

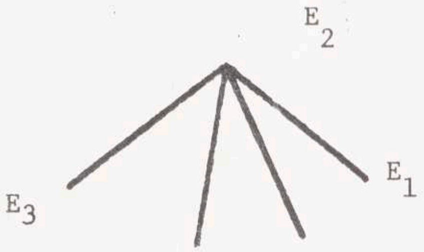
The datum for an X may also be in the form (E_4 E_5 E_6 E_1 E_2 E_3).

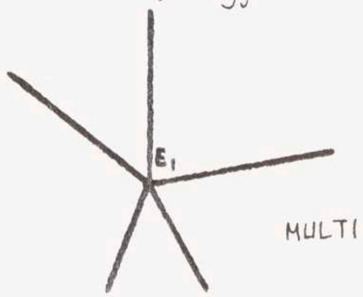
Vertices of four lines which are not of type K or X are either of type PEAK or MULTI.

Other types of vertices.

PEAK.- Formed by four or more lines, when there is an angle bigger than 180°

PEAK





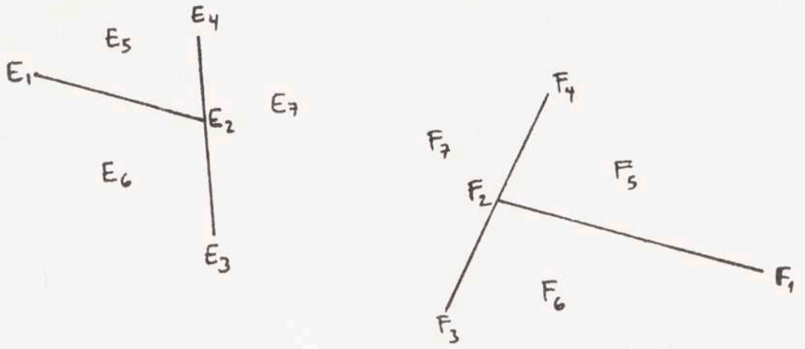
MULTI.- Vertices formed by four or more lines, and not falling in any of the **preceding types, belong to the type** MULTI. R3 **contains no PEAKS** or MULTIS.

The datum for vertices of type PEAK is of the form (E_1 E_2 E_3), where E_2 is the region that contains the angle bigger than 180 degrees; E_1 is the vertex before E_2, and E_3 is after (in the +↺ sense).

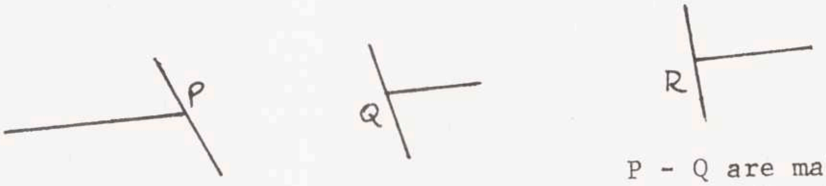
The datum for vertices of type MULTI is of the form E_1, where E_1 is the vertex itself.

<u>NEXTEs or Matching T's</u> Two T's which are colinear and facing each other (see figure) are called "matching T's," and each one is the "nexte" of the other. The indicator "NEXTE" is placed in such vertices.

If the region E_7 of a T (see figure) is the background, that T can not be a matching T.

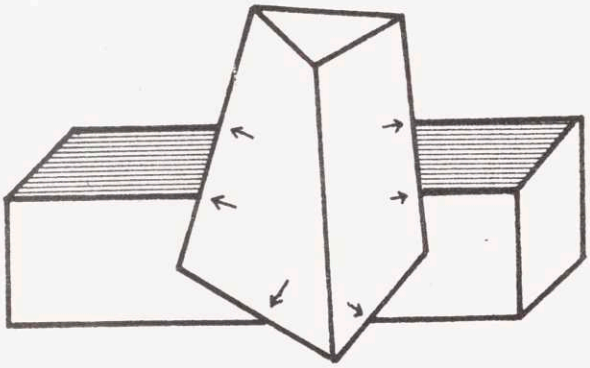


In the figure, E_2 and F_2 are matching T's because E_1-E_2 is colinear with F_2-F_1. It is not required of E_3-E_4 to be parallel to F_3-F_4. If several pairs of T's are possible, the closest is chosen:

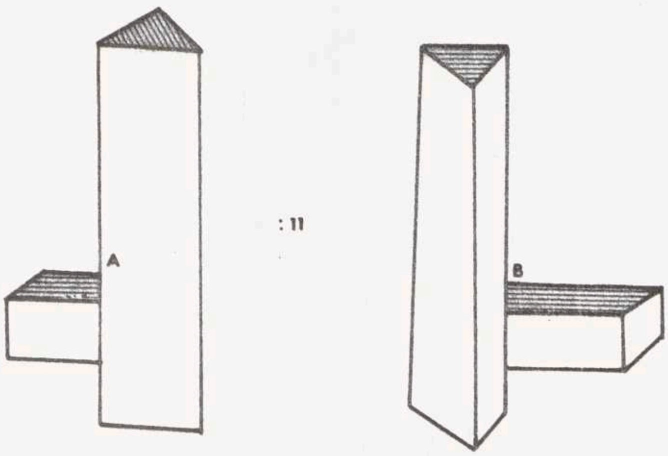


P - Q are matching T's, and not P - R.

The matching T's will get involved in the determination of places where a body is occluded by another object and later emerges visible again.



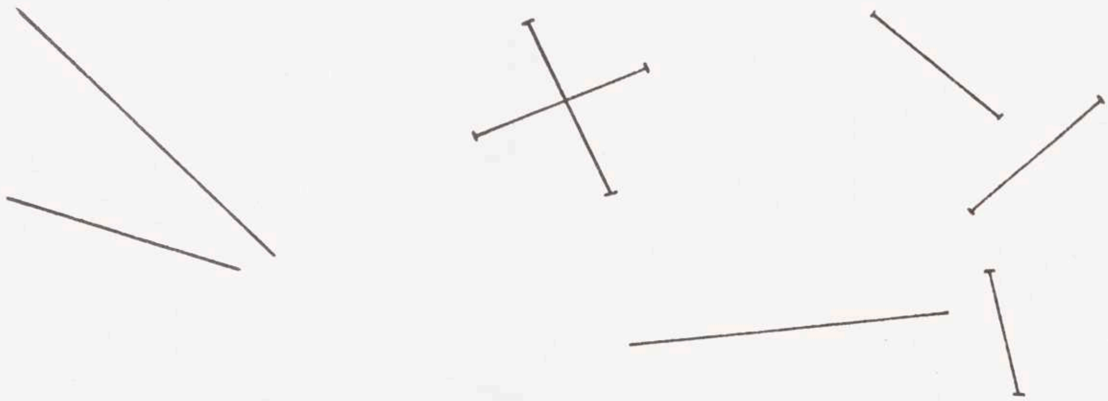
For two T's to be NEXTEs or matching T's, it is required that neither E_7 nor F_7 be background. This requirement should be extended to all regions between E_7 and F_7, since a line can not go "under" the background region:



A and B can not be NEXTEs, since :11 is the background.

SUGGESTION

Two straight lines always intersect (possibly at infinity); a way to detect these background regions is to write functions (subroutines) that find out if two segments of line intersect, or if one segment intersects with a line.



LINES AND SEGMENTS

In the plane, two straight lines always meet.
Two segments, or a line and a segment, may or may not meet. (a segment is a finite portion of a line).

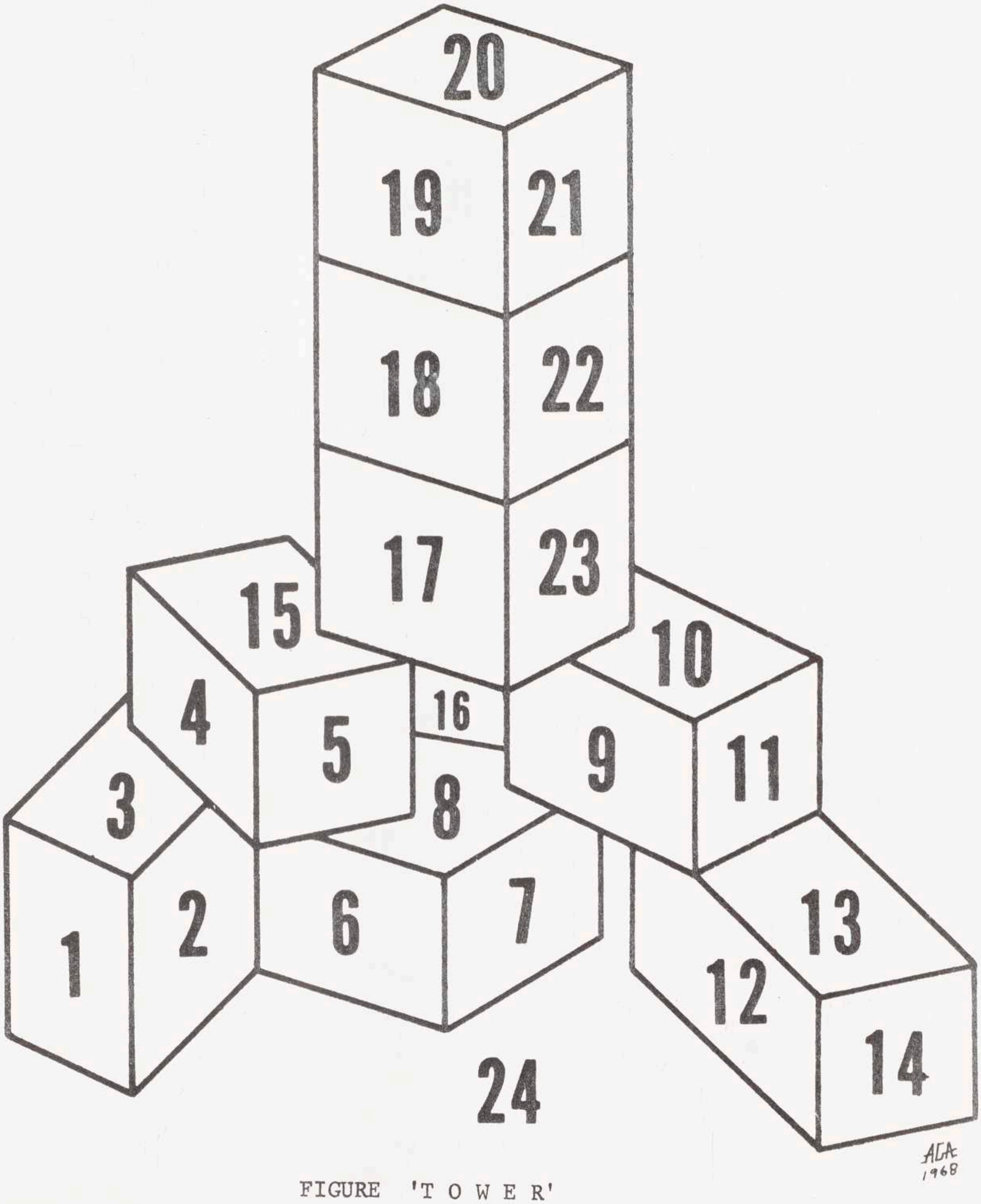
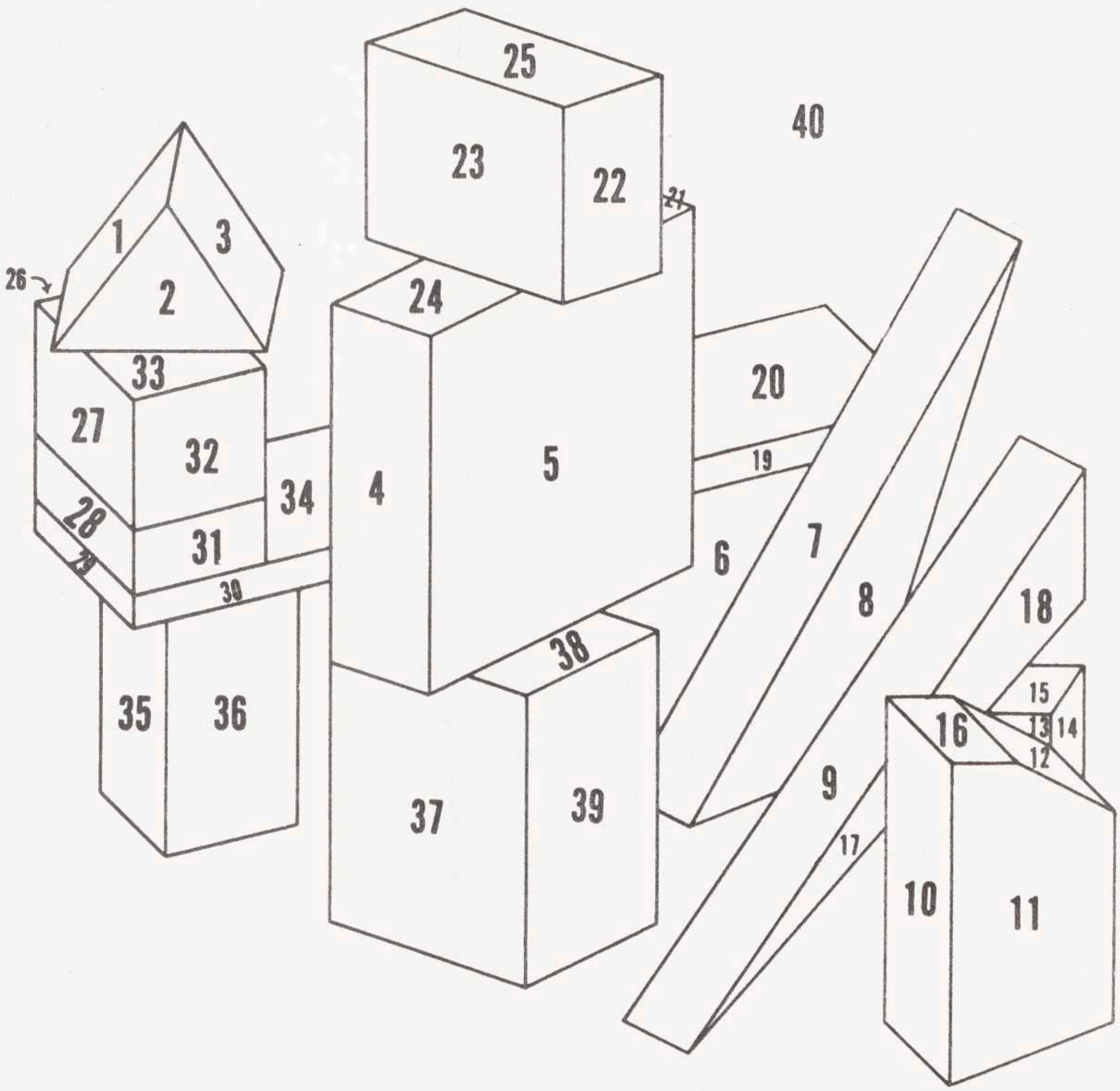


FIGURE 'T O W E R'



25
40
23
22
21
1
3
26
2
24
33
20
27
32
5
19
34
4
28
31
7
6
29
30
8
18
38
15
35
36
16
13
14
12
9
37
39
17
10
11

FIGURE 'M O M O'

T H E   P R O G R A M

We now describe SEE, and how it achieves its goals, by discussing the procedures, heuristics, etc., employed and the way they work. We begin with several examples.

Example A. Scene 'TOWER'. This scene (see figure 'TOWER') is analyzed by SEE, with the following results:

```
RESULTS
(BODY 1. IS :2 :3 :1)
(BODY 2. IS :15 :5 :4)
(BODY 3. IS :23 :17)
(BODY 4. IS :6 :7 :8)
(BODY 5. IS :10 :11 :9)
(BODY 6. IS :13 :14 :12)
(BODY 7. IS :18 :22)
(BODY 8. IS :20 :19 :21)
```

Results for scene TOWER

Example B. Scene 'MOMO'. Details of the program's operation are given. (skip to next page, if you wish).

↑Z $L SEE 1↩	Go to DDT and load file SEE 1 (in tape GUZMAN F), a binary dump of the program SEE.
$G	Start.
(UREAD MOMO S1 3) ↑Q	Read the file MOMO S1 (in tape GUZMAN C) from tape drive 3.
(PREPARA MOMO)	Convert MOMO from its Input Format form to Internal Format, the proper form that SEE expects.
(SEE (QUOTE MOMO))	Call SEE to work on MOMO.

Results appear in next page.

Notes: ↑Z (control Z) is keyed by striking the Z key while holding down simultaneously the CONTROL key. (A.I. Memos 161,157,149)

↩ denotes carriage return.

$ denotes the character "alt. mode". (see also instructions in listing)

```
SEE 58 ANALYZES MOMO
EVIDENCE
LOCALEVIDENCE
TRIANG
GLOBAL
((NIL) ((:38) G0044 G0043 G0041 G0040) ((:19) G0046 G0045 G etc
LOCAL
(LOCAL ASSUMES (:17) (:9) SAME BODY)
(LOCAL ASSUMES (:9 :17) (:18) SAME BODY)
LOCAL
((NIL) (NIL) ((:6)) (NIL) (NIL) (NIL) ((:38 :37 :39) G0043 etc.
LOCAL
(((:3 :2 :1) G0081 G0029 G0030 G0028) ((:32 :33 :27 :26) G0etc.
LOCAL
SMB
RESULTS
(BODY 1. IS :3 :2 :1)
(BODY 2. IS :32 :33 :27 :26)
(BODY 3. IS :28 :31)
(BODY 4. IS :20 :34 :19 :30 :29)
(BODY 5. IS :36 :35)
(BODY 6. IS :24 :5 :21 :4)
(BODY 7. IS :25 :23 :22)
(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)
NIL
```

RESULTS FOR MOMO

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 pre-processor will tend to make this kind of error.

Example C. R3. Analysis by SEE gives

```
(BODY 1. IS %:2 %:1 %:4)
(BODY 2. IS %:6 %:5 %:3)
```

RESULTS FOR 'R3'

The % sign indicates the dextral scenes (cf. page 233). The signs may be ignored.

The Parts of SEE The program is straightforward; it does not call itself recursively; it does not do "pattern matching"; it does not do tree search. It is formed by several main parts, sequentially executed. They are

LINKS FORMATION. 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).

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 evidences.

NUCLEI CONSOLIDATION. 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).

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.

BODY RETOUCHING. 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. If necessary, more nuclei consolidation could be done after this step.

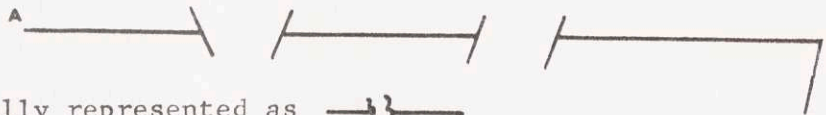
A last attempt is done to associate the remaining single regions to other bodies.

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

Auxiliary Routines

Three functions are used constantly, and will be described now.

THROUGHTES "Through a chain of T's." Allows properties or configurations to extend along straight lines; for instance, the property «'A' has as neighbor an L»  can be extended so as to say «throughtes, 'A' has as neighbor an L».



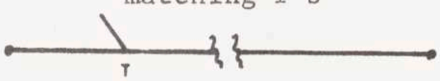
schematically represented as 

Strict definition.  is defined as one of

(1) • (meaning the two vertices in both sides of are in fact the same).

(2) 

matching T's

(3) 

(4) 

Example of 

See also annotations on listing.

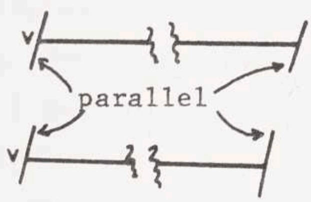
GOODT If a vertex V is considered a "good T", (GOODT V) is TRUE; false otherwise.

(GOODT V) = F if V is not a "T"

F if 

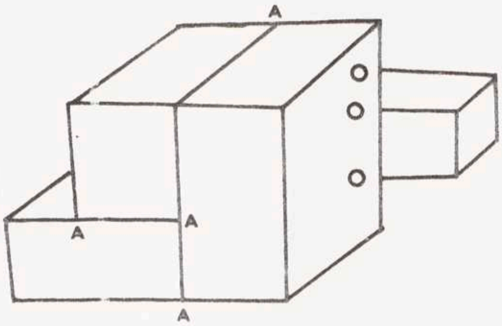
T if V has a NEXTE.

F if

F if 

T otherwise.

As we see, this function tries to distinguish between T's originated by occlusion, such as O, and T's originated by accident (A).

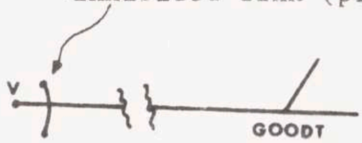


NOSABO "Not same body." Acts as a link inhibitor.

If consulted, (NOSABO .. V ..) will inhibit, in the following conditions, the link that vertex V may have created:

(1) 

inhibited link (prohibited, ignored, forbidden, not created)

(2) 

(3) 

(4) 

(5) 

Nosabo tries to find conditions indicating that two regions should not be considered as part of the same body; hence, if consulted, Nosabo may forbid a link among them. Some heuristics place links without asking Nosabo's aproval and Nosabo can not "erase" a link placed without its authorization.

If none of conditions (1) to (5) is met, Nosabo will be False, indicating no inhibition was found, and it is up to the program that asked Nosabo's opinion to lay or fail to lay the link in question.

We proceed now to explain in considerable detail each of the parts of SEE. This will help the reader to understand the behavior of the program, its strength and deficiencies.

LINK FORMATION

Several subroutines are devoted to creating weak and strong links. See also listing.

CLEAN Removes several unwanted properties.

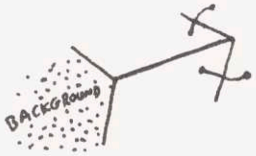
EVERTICES Each vertex is considered under the following rules:

L.-  No evidence is created directly by this type of vertex. Nevertheless, the "L" is used in many combinations with other vertices to account for evidence. As we saw, Nosabo uses L's. "Legs" will use them, too.

FORK.- 

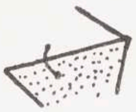
== No link is created if any of the three regions is background (but see below).
Example (unless otherwise indicated, all examples are from figure 'BRIDGE'): Vertex J does not generate links.

== Otherwise, three links are created as shown, except that each one may be inhibited by Nosabo.
Example. Vertex JB only produces link :5-:8.
Link :5-:9 is inhibited because S is a 'T'; Nosabo also forbids link :8-:9 because KB is an 'arrow'.
This is the most powerful of the heuristics.

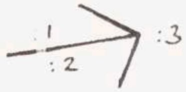


== Two links are created as shown, without asking Nosabo, if the fork is connected to the central line of an arrow. (No link is put here )
Example: In fig. R19, PA generates links :29-:17 and :35-:17.

This last heuristic is of help where there are concave objects (Fig. R19).

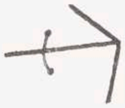
ARROW.-  == Link if an L is connected to its central line, and the region shaded contains only that arrow as a "proper-arrow," and no Forks.

Region :1 contains arrow A as a "proper-arrow"; also region :2, but not region :3. Capisce?



Example. BB links:10 with :4.

Allows "lateral faces" of legs to be properly identified and agglutinated.

 == Otherwise, link except if inhibited by Nosabo.

Example. D lays a link between :26 and :23.

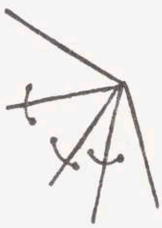
Powerful and general heuristic.

X.-  == No link if the X comes from the intersection of two lines.

 == Otherwise, link as shown except if Nosabo disagrees.

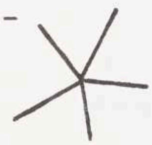
Example. G originates links :26-:22 and :21-:30; this last one will later be erased or disregarded, since :30 is the background.

K.-  == No link.

PEAK.-  == Links are established between contiguous regions, except those to the region containing the angle bigger than 180 ^{o}. These links are subject to Nosabo inhibition.

Example. In fig. 'CORN', JJ generates links :8-:9 and :9-:10.

Of certain use, specially with pyramids and "pointy" objects.

MULTI.-  == No link.

The reason is:

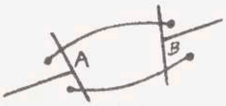
(1) if the vertex is "genuine" (cf. page 44),

although it generates no links, the object having it will probably possess many other vertices, through which links will get established, and

(2) if the vertex if "false" because is the result of the casual coincidence of two or more genuine vertices, mistakes are avoided by abstaining of generating links. This is generally the case. An improvement is possible, by allowing MULTI vertices to place links.

SUGGESTION

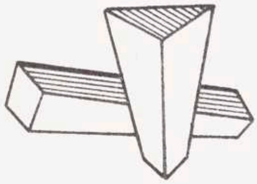
T.-

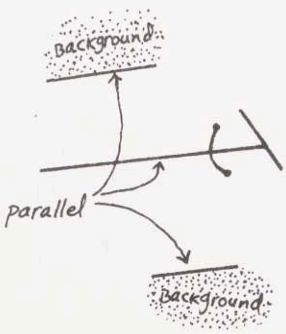


== If matching T's, link as shown, without consulting Nosabo, **Avoid linking to the background.** Each pair of matching T's produces these links only once; that is, we do not produce two links while analyzing A and another two at B.

== Do not link if the middle region of a 'T' is the background.

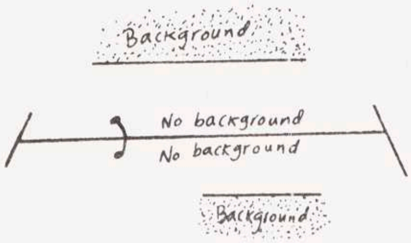
What we are trying to do here is to find places where a body **appears as two disconnected parts.**





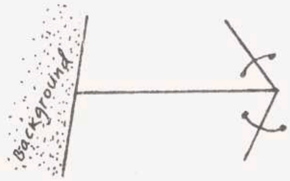
== Link (without Nosabo's consent) as shown if the central segment of the 'T' separates two non-background regions, and these have the background as neighbor, and part of the separations between background and no-background are parallel to the central segment of the 'T'.

Avoid double links in the following case (link just once):



Example. TA links :21 with :27 (F-G, RA-TA and JA-IA are parallel).
Favors occluded bodies with parallel faces.

== Also, see "STUDY" in listing, still an experimental feature.



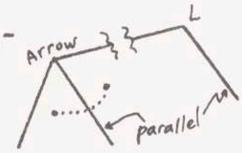
== Two links are placed as shown (without asking Nosabo) if the central line of the T is connected to the central line of an arrow.
It is of help where there are concave objects.

Table 'Global Evidence' shows compactly the main rules just discussed.

LOCALEVIDENCE Weak or local links are laid here; they are used to indicate, in a feebler way, that two faces or regions may be part of the same object.

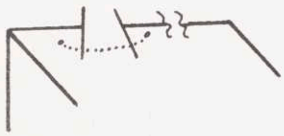
Nosabo can not inhibit local links.

LEG.-



== A weak link is placed as shown (dotted) if, Throughtes, an L is connected to an Arrow, and the two indicated edges are parallel.

We call this configuration 'Leg'.
Example (all examples from figure 'BRIDGE', except if counterindicated). Vertex FA is a Leg (FA - QB is parallel to EA - DA) that links weakly :18 with :19.



== In a Leg, if there are two matching T's as shown, a weak link is placed correspondingly.
Example. In fig. 'TRIAL' (page 88), a weak link or evidence is placed between :7 and :4, because EE is a Leg, and L and E are matching T's.

The heuristics described will sometimes produce a "wrong linkeage," linking two regions that do not belong to the same body. These mistakes are not likely to confuse SEE, since the handling of these links (and all of SEE, in general) is done under the assumption or knowledge that the information is noisy and somewhat unreliable.

Strong links are shown dotted; weak links are not shown.

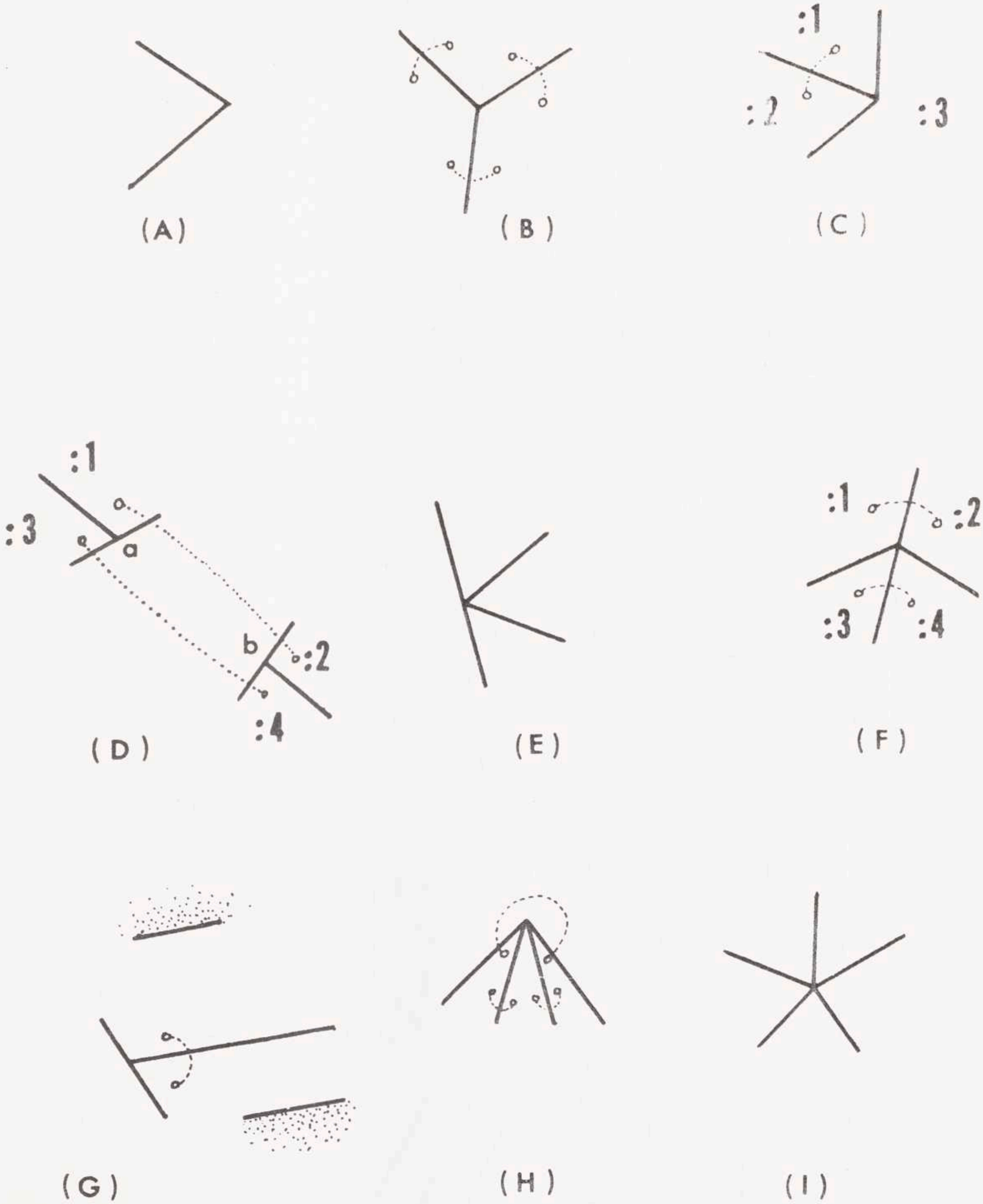
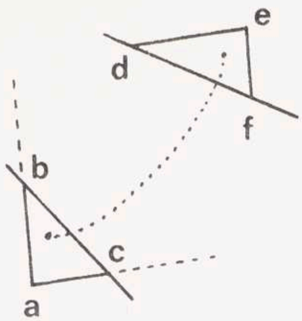


TABLE 'GLOBAL EVIDENCE'

TRIANGLE.- == A Triangle is a 3-vertex region, of which two are interconnected T's, the type of the other vertex being irrelevant.



Two triangles are weakly linked if they are

(1) "facing each other", and

(2) "properly contained", meaning that D has to fall on the same side of AB as C does, and similarly for the other vertices, and

(3) AB is parallel to EF, and AC to DE.

The heuristic helps with faces of a prism that is badly obscured. It does not help much, since it gives only a weak link. On the other hand, this weakness prevents mistakes when the two griangles are not from the same body.

SUGGESTION

A possible improvement consists of choosing the closest of two triangles, if several candidates are possible.

Example. In figure 'WRIST' (page 136), weak links are placed between triangles 5 and 6, and between 1 and 2.

Example. Figure 'TRIAL' receives the following strong links (full lines) and weak links (dotted lines)

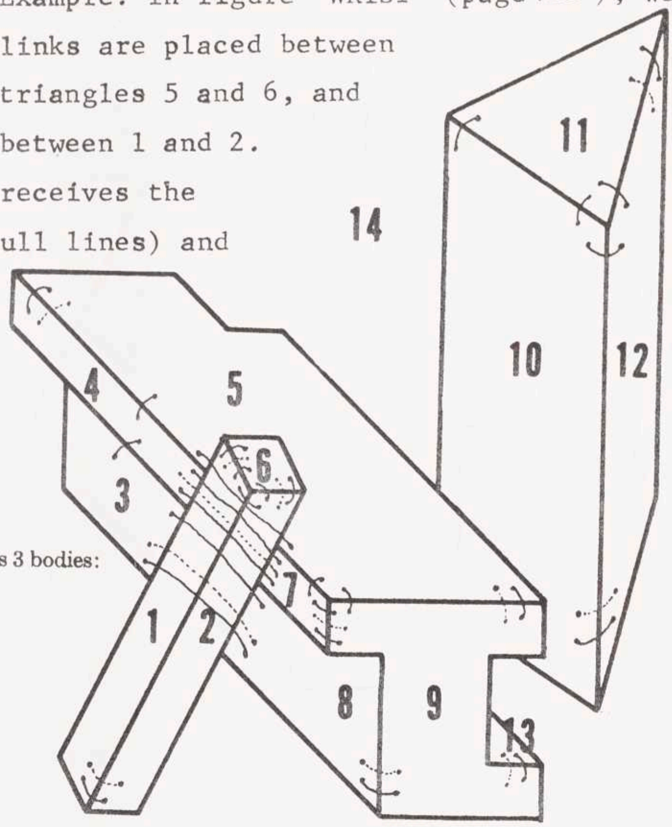


FIGURE TRIAL

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)

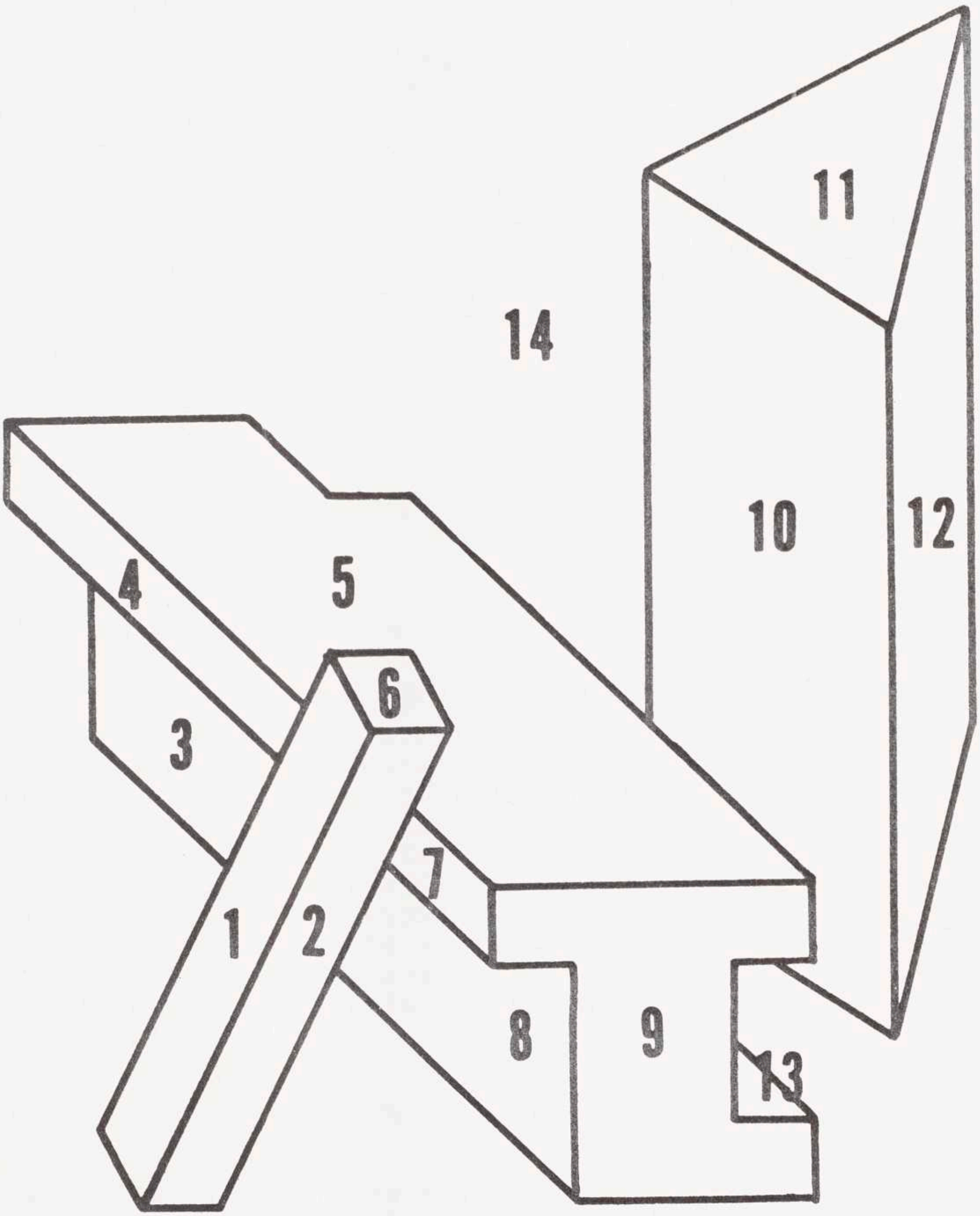


FIGURE 'TRIAL'

The links could be represented as

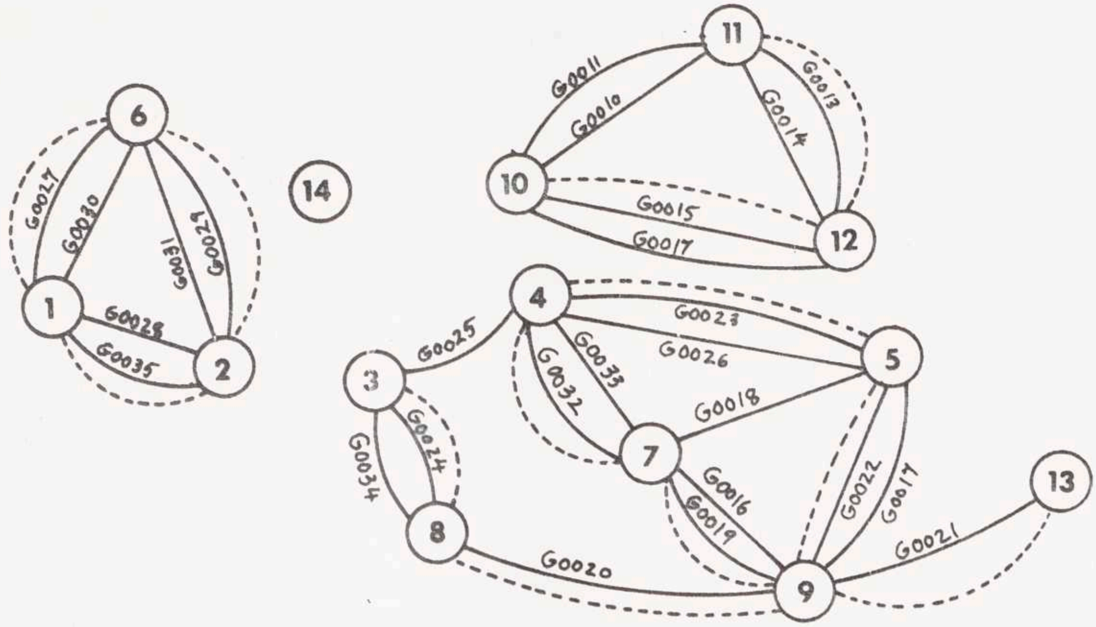


Figure 'TRIAL - LINKS'

Strong (solid) and weak (broken lines) links of figure 'TRIAL'.

SEE prints these links in the following way:

:11 has four *strong* links emanating from itself.

```
((NIL) ((:11) G0014 G0013 G0011 G0010) (
(:12) G0015 G0014 G0013 G0012) ((:13) G0
021) ((:9) G0022 G0021 G0020 G0019 G0017
 G0016) ((:10) G0015 G0012 G0011 G0010)
((:3) G0034 G0025 G0024) ((:4) G0033 G00
32 G0026 G0025 G0023) ((:6) G0031 G0030
G0029 G0027) ((:5) G0026 G0023 G0022 G00
18 G0017) ((:7) G0033 G0032 G0019 G0018
G0016) ((:8) G0034 G0024 G0020) ((:2) G0
035 G0031 G0029 G0028) ((:14)) ((:1) G00
35 G0030 G0028 G0027))
```

Strong Links of 'TRIAL'

Weak links of scene 'TRIAL' are

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

Weak links of 'TRIAL'.

There is a weak link between :12 and :10

The next step is to gather all this evidence and to form tentative hypotheses of objects as assemblages of faces with many links among them.

NUCLEI CONSOLIDATION

All the links to the background are deleted, since it can not be part of any body.

Strong and weak links exist among the different regions of a scene. They are consolidated in that order by two subroutines, Global and Local.

<u>GLOBAL</u> Groups of faces with an abundance of strong links among them are first found; these "nuclei" will later compete for other faces more loosely linked.

<u>Definition</u>: a <u>nucleus</u> (of a body) is either a region or a set of regions that has been formed by the following rule.

<u>Rule</u>: If two nuclei are connected by two or more strong links, they are merged into a larger nucleus.

More detailed rules appear in page 25 , in section 'Simplified view of Scene Analysis'.

For instance, in the figure below, regions :1 and :2 are put

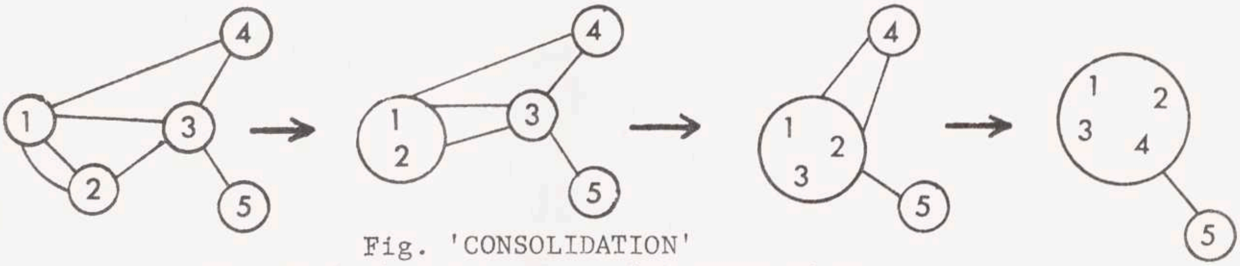


Fig. 'CONSOLIDATION'
Two links between two nuclei merge them.

together, because there exist two links among them, to form nucleus :1-2. Now we see that region :3 has two links with this nucleus :1-2, and therefore the new nucleus :1-2-3 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 is at most one link. For example, figure 'TRIAL-LINKS' will be transformed into figure 'TRIAL-NUCLEI'.

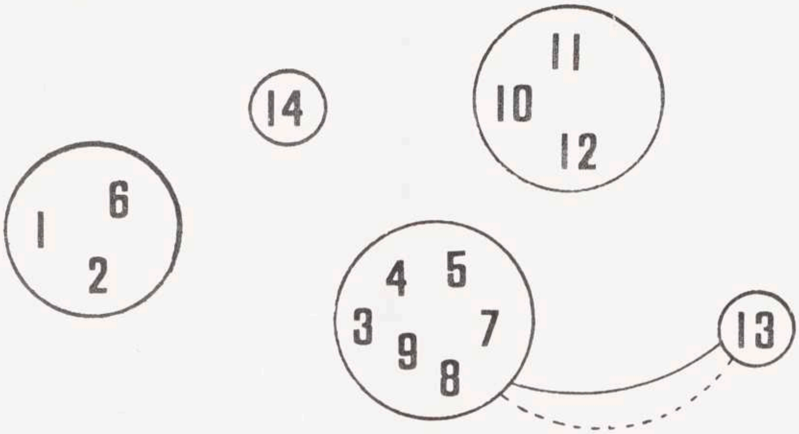


Figure 'TRIAL - NUCLEI'
Maximal nuclei of scene TRIAL.

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

The weak links of figure TRIAL are shown as dotted lines in figure 'TRIAL-LINKS' (page 90); they transform figure 'TRIAL-NUCLEI' into figure 'TRIAL-FINAL'.

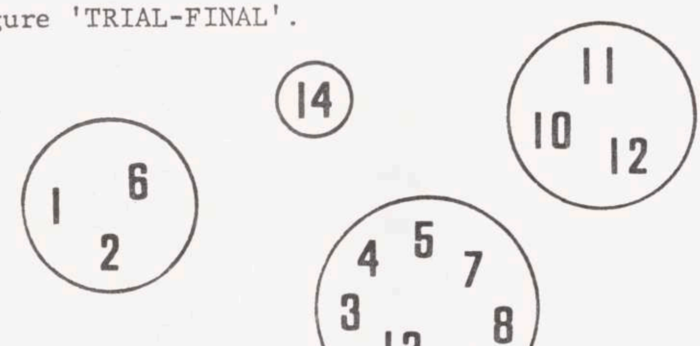


Figure 'TRIAL - FINAL'
Nuclei of scene TRIAL after merging suggested by local links.

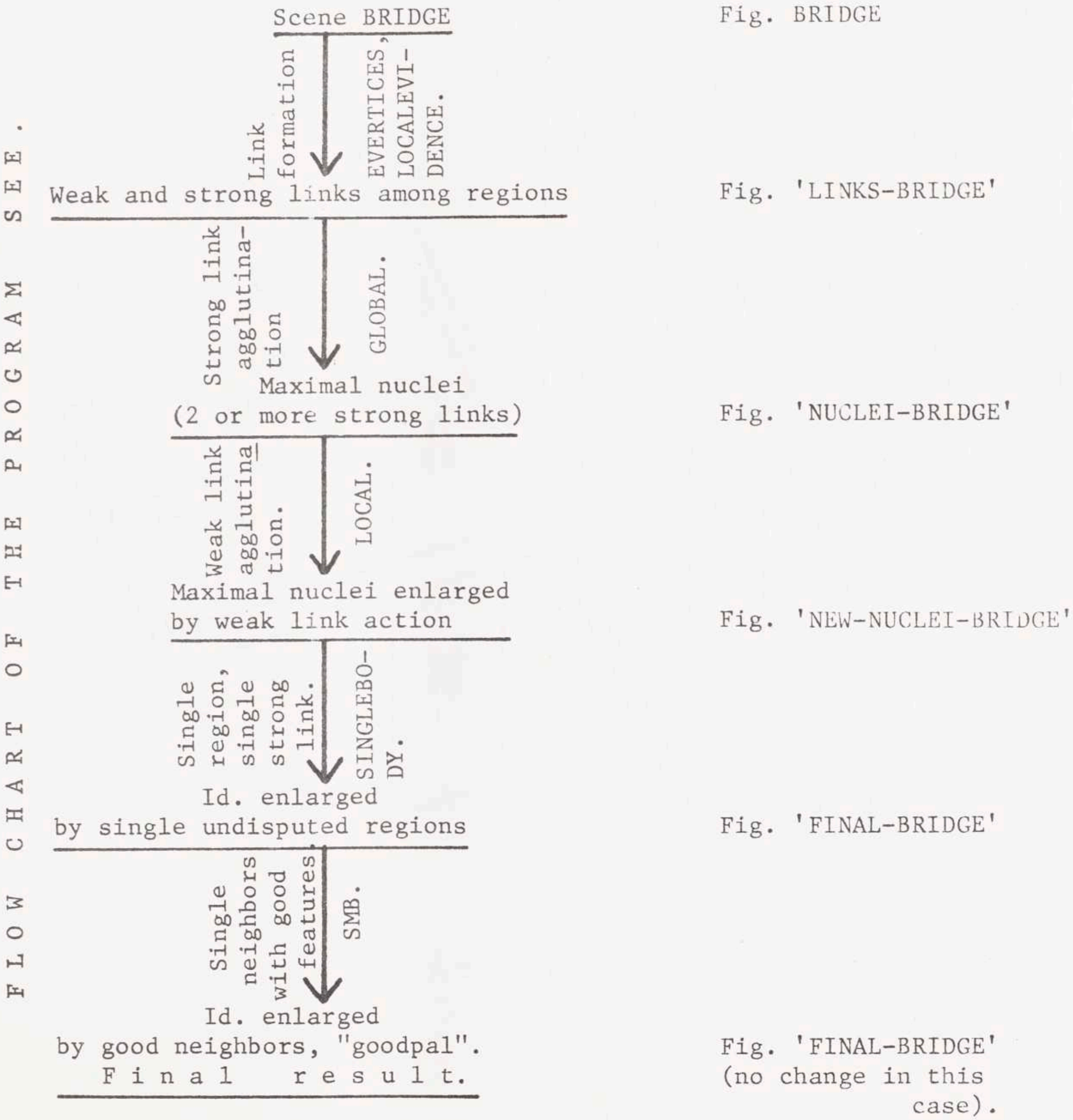
BODY RETOUCHING

Additional heuristics assign unsatisfactory faces to existing nuclei, or isolate them. SINGLEBODY and SMB are used for this task.

SINGLEBODY A strong link joining a nucleus and another nucleus composed by a single region is considered enough evidence to merge the nuclei in question if there is no other link emanating from the single region. A message is printed indicating these merges.

Such rules produce no change in fig. 'TRIAL-FINAL', and therefore its nuclei will be reported as bodies.

A more complex example shows the retouching operation. Figure 'BRIDGE' undergoes these transformations:



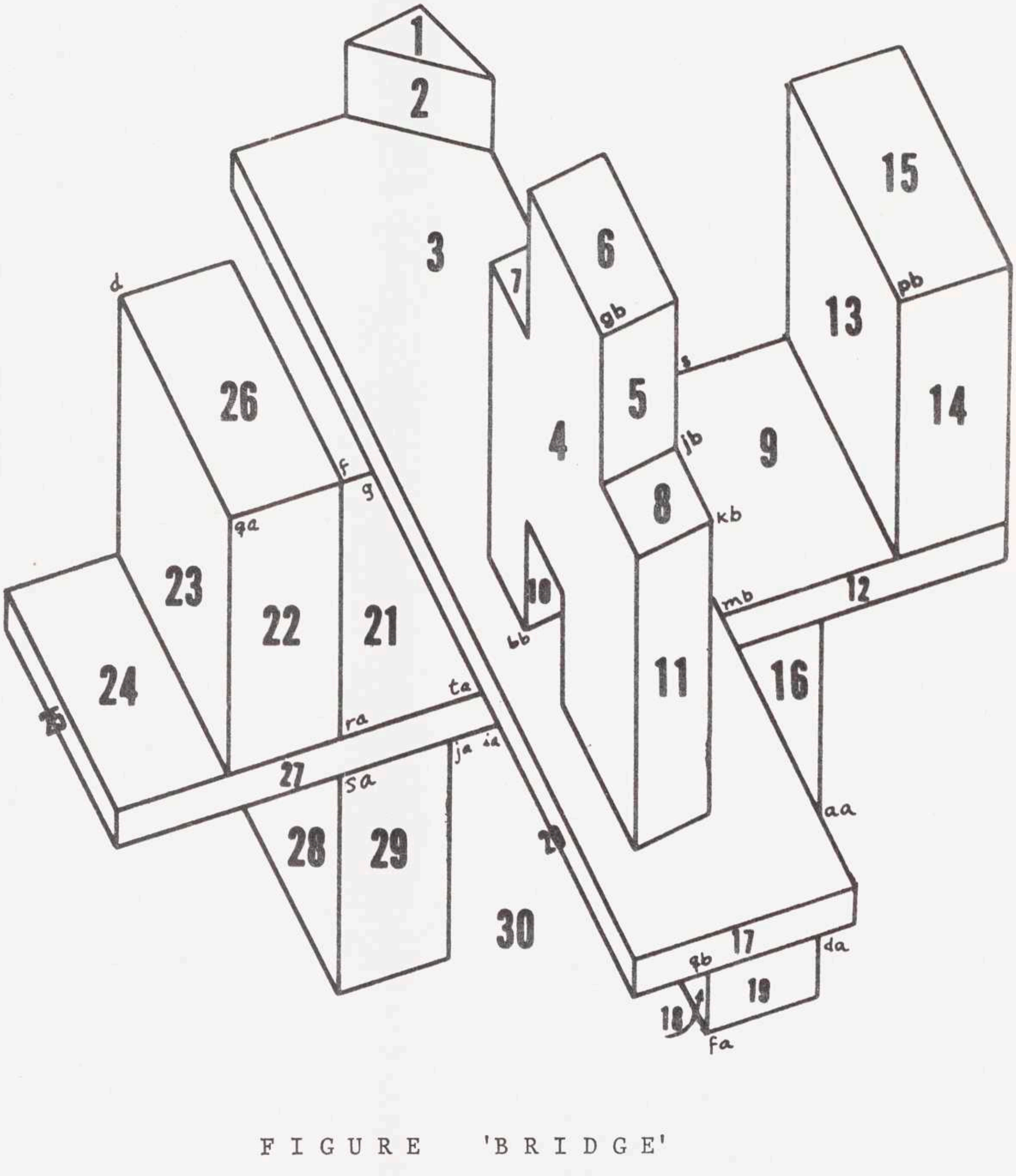


FIGURE 'BRIDGE'

FIGURE 'LINKS-BRIDGE'

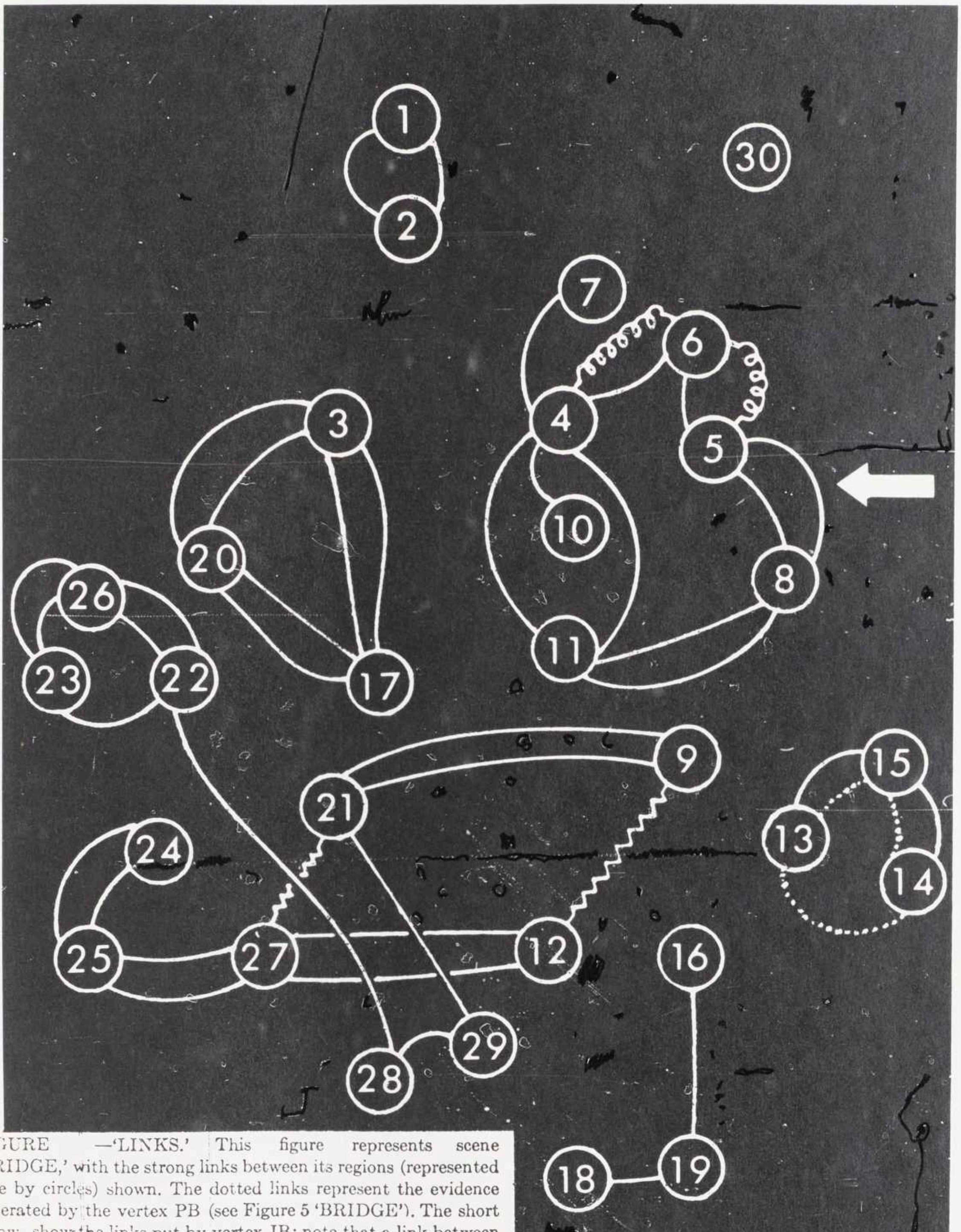


FIGURE —'LINKS.' This figure represents scene 'BRIDGE,' with the strong links between its regions (represented here by circles) shown. The dotted links represent the evidence generated by the vertex PB (see Figure 5 'BRIDGE'). The short arrow shows the links put by vertex JB; note that a link between :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. Link between :8 and :9 is also inhibited. 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 'NUCLEI - BRIDGE'

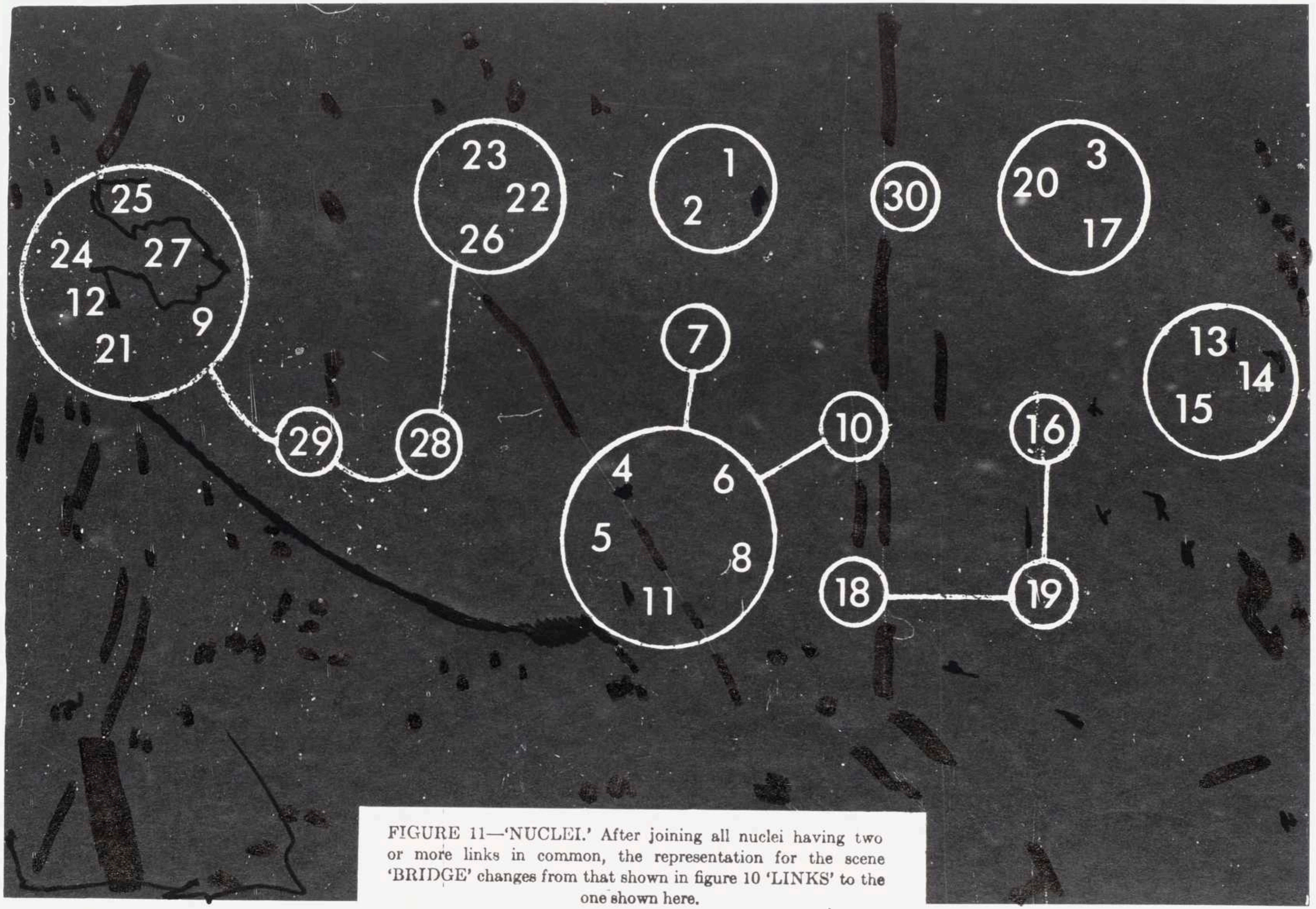


FIGURE 11—'NUCLEI.' After joining all nuclei having two or more links in common, the representation for the scene 'BRIDGE' changes from that shown in figure 10 'LINKS' to the one shown here.

FIGURE 'NEW-NUCLEI-BRIDGE'

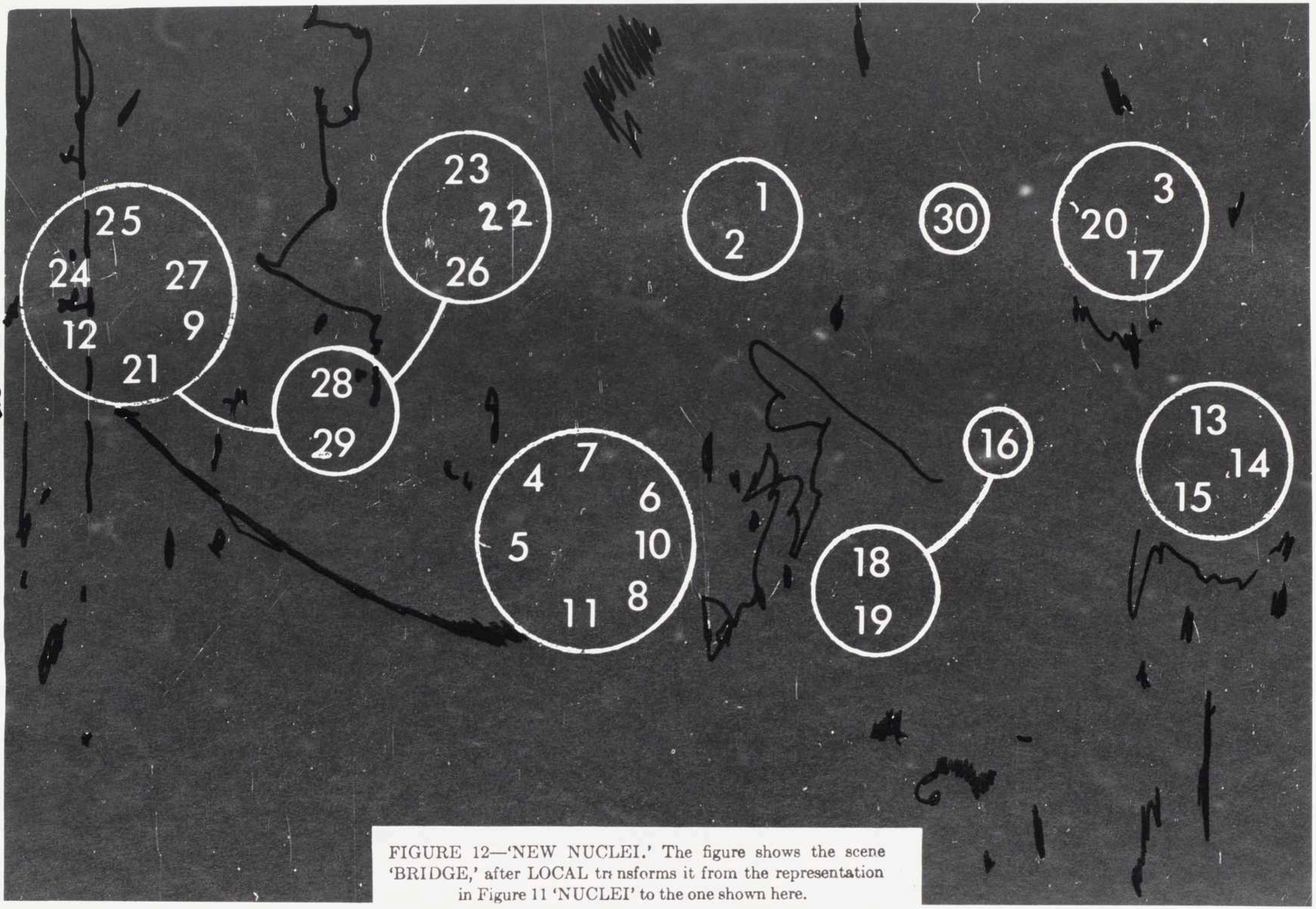


FIGURE 12—'NEW NUCLEI.' The figure shows the scene 'BRIDGE,' after LOCAL transforms it from the representation in Figure 11 'NUCLEI' to the one shown here.

FIGURE 'FINAL - BRIDGE'

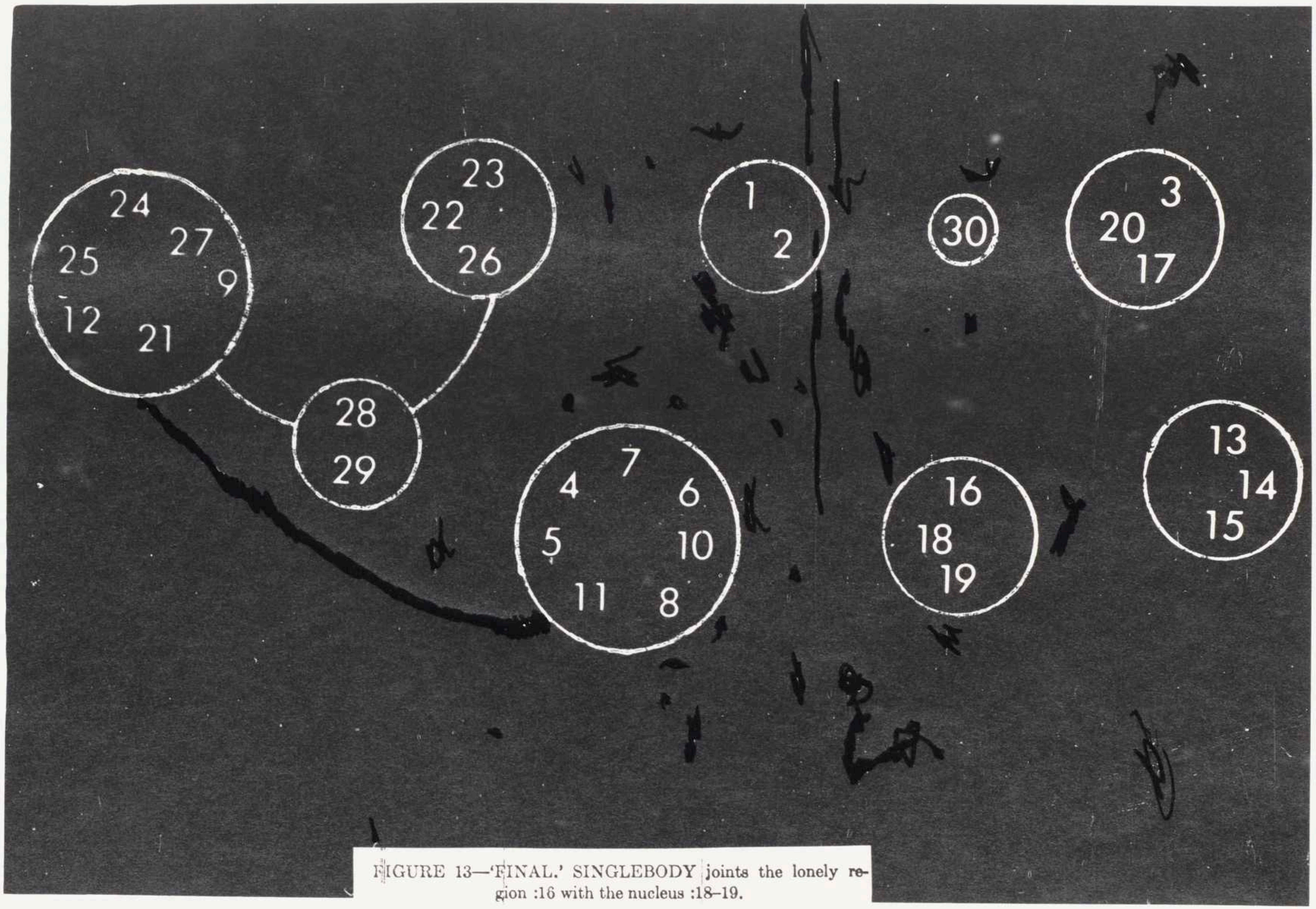


FIGURE 13—'FINAL.' SINGLEBODY joints the lonely region :16 with the nucleus :18–19.

We see that in figure 'NEW-NUCLEI-BRIDGE', nucleus :16 is merged by SINGLEBODY with nucleus :18-19 (see figure 'FINAL-BRIDGE'). 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, still will not be merged, since two links emerge from it: two nuclei claim its possession.

This rule joins single regions having only one possible "owner" nucleus.

<u>SMB</u> Two systems of links are used by SEE. One consists of weak and strong links, produced by examining each vertex, and culminates forming nuclei under GLOBAL, LOCAL, etc.

The second system constitutes a different network of links; SMB works in the second system. It is motivated by the desire to collect evidence not directly available through the vertices. It gathers evidence from the <u>lines</u> or <u>boundaries</u> separating two regions, in an effort to answer the question: Are two given neighboring regions part of the same object, or are not they? That is, are two contiguous regions "good neighbors" ("good·pals")? If they are, a special link, <u>s·link</u>, is placed, eventually forming a network independent of weak and strong links, that will collapse in a somewhat peculiar way. Thus, a great amount of unnecessary duplication could be possible in the information carried by both systems of links. To reduce it, the s·links are designed to complement and extend, rather than to re-do, the agglutination produced by weak+strong links. They (the s·links) will, therefore, mainly study single faces not satisfactorially accounted for.

SMB uses the predicate (GOODPAL R S), which acquires the value T (true) if R and S are two contiguous "good neighbors" regions. To satisfy this, their common boundary must not be empty, and must lack L's, FORKs, ARROWs, K's, X's, PEAKs, MULTIs. In addition:

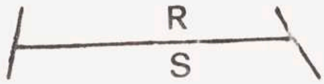
== Not good: (GOODPAL R S) = F

== Not good: (GOODPAL R S) = F

"L" or (in general) vertex that makes (NOSABO R S) to be true.

== O. K. otherwise: (GOODPAL R S) = T.

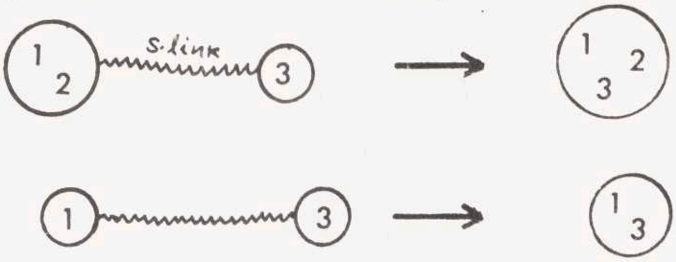
In particular,

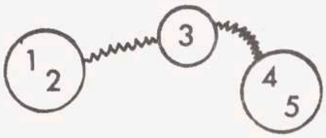


is O. K. if (NOSABO R S) = F.

SMB analyzes the nuclei formed under weak+strong links that, after SINGLEBODY actuation, still remain formed by a single face or region. The steps are:

1. A network of s·links is formed by putting a s·link between regions forming a nucleus all by themselves, and their goodpal neighbors.
2. If to one of those regions exactly one nucleus is s·linked (that is to say, if such single-region single-nucleus has precisely one good·pal), the region gets absorbed by the nucleus; otherwise it is reported as a body consisting of a single region.





does not change because :3 has two slinks.

Note that

a. The slinks are not used to form nuclei as the weak+strong links were; they only help certain isolated faces to join bigger structures.

b. Two slinks between two regions have the effect of one.

Example. In figure 'HARD', regions :6 and :7 get joined by SMB.

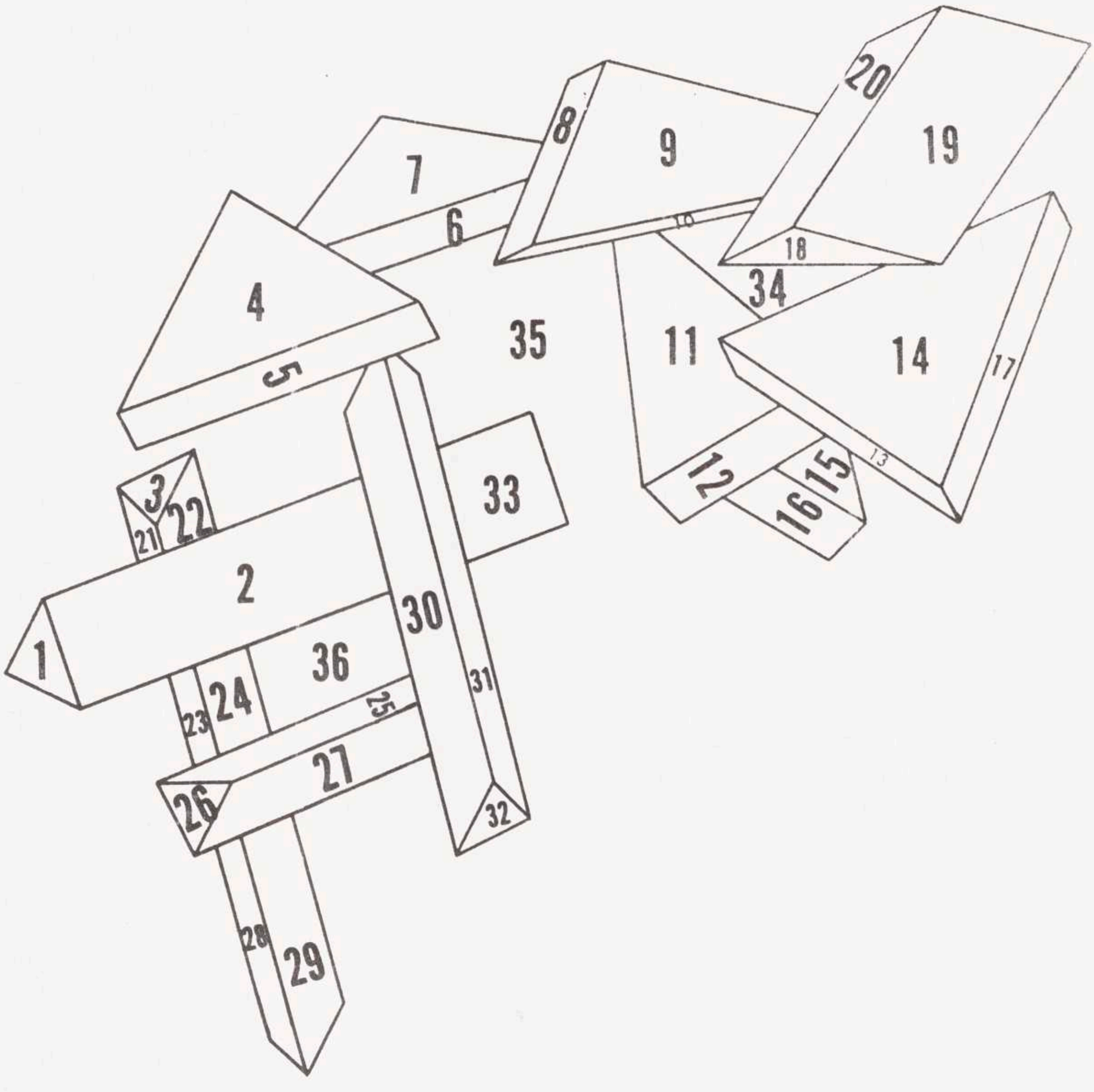


FIGURE 'H A R D'

This scene shows the use of SMB.

```
SEE 58 ANALYZES HARD
EVIDENCE
LOCALEVIDENCE
TRIANG
GLOBAL
((NIL) ((:34)) ((:6)) ((:36)) ((:24) G0026 G0025 G0023 G0 etc.
0044 G0043 G0042) ((:17) G0047 G0046 G0045 G0044) ((:7)) etc
0041 G0039) ((:21) G0050 G0040 G0039 G0029 G0028 G0027) ( ...
0038 G0036 G0019) ((:26) G0054 G0053 G0037 G0036) ((:27) ...
 G0055 G0023 G0020 G0015) ((:32) G0057 G0056 G0034 G0033)
8 G0048) ((:4) G0058 G0048) ((:10) G0059 G0032 G0031) ((:
:19) G0064 G0063 G0062 G0061) ((:20) G0064 G0062 G0060 G0
:30) G0056 G0035 G0033 G0016) ((:15) G0066) ((:16) G0066)
((NIL) ((:34)) ((:6)) ((:36)) (NIL) (NIL) (NIL) (NIL) ((:
019 G0053 G0036 G0054 G0038 G0037 G0019) (NIL) ((:24 :22
0040 G0039 G0029 G0028 G0027 G0024 G0022 G0055 G0023 G002
) (NIL) ((:5 :4) G0048 G0058 G0048) (NIL) ((:13 :17 :14)
:18 :19 :20) G0060 G0064 G0063 G0061 G0064 G0062 G0060 G0
:32 :31 :30) G0033 G0057 G0034 G0056 G0035 G0033 G0016) (

LOCAL
(LOCAL ASSUMES (:11) (:12) SAME BODY)
(LOCAL ASSUMES (:15) (:16) SAME BODY)
((NIL) ((:34)) ((:6)) ((:36)) (NIL) (NIL) ((:7)) (NIL) (N
019) ((:24 :22 :3 :23 :21 :28 :29) G0020 G0026 G0025 G004
0055 G0023 G0020 G0015) ((:1 :2 :33) G0052 G0051 G0017 G0
43 G0047 G0046 G0044 G0047 G0045 G0043 G0042) (NIL) ((:18
 :10 :8) G0032 G0032 G0065 G0059 G0031 G0030) ((:32 :31 :
) (NIL) ((:35)) ((:12 :11) G0067) (NIL))
LOCAL
(((:12 :11) G0067) ((:16 :15) G0066) ((:32 :31 :30) G0033
G0065 G0059 G0031 G0030) ((:18 :19 :20) G0060 G0064 G0063
6 G0044 G0047 G0045 G0043 G0042) ((:5 :4) G0048 G0058 G00
3 :21 :28 :29) G0020 G0026 G0025 G0049 G0041 G0021 G0050 (
15) ((:25 :26 :27) G0019 G0053 G0036 G0054 G0038 G0037 G0
LOCAL
SMB
(SMB ASSUMES :7 :6 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)
(BODY 7. IS :5 :4)
(BODY 8. IS :1 :2 :33)
(BODY 9. IS :24 :22 :3 :23 :21 :28 :29)
(BODY 10. IS :25 :26 :27)
(BODY 11. IS :7 :6)
NIL
```

RESULTS FOR HARD

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 'FINAL-BRIDGE', for instance, as non-existent. These links are the result of imperfections in the heuristics, mistakes in the placement of links, and may point out different parsings. An improvement to SEE will be to try to "explain" these residual links.

<u>Summary</u> 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.

Examples and discussion are in next section.

ANALYSIS OF MANY SCENES

Until we have an adequate analytic theory, the behavior of a heuristic program is best understood with examples. There are several ways to go about this:

<u>Simple</u> In order to learn what a program does, simple examples, each one illustrating a single feature or group of features, are very appropiate.

<u>Favorable</u> A shiny impression of a set of routines is obtained by presenting 'favorable' cases, designed to enhance the characteristics of the program in front of the unsophisticated observer.

Of course, of all possible inputs, there is a subset that will produce outputs very pleasant in terms of speed, easiness of programming, generality, accuracy, or whathever other feature that system advertises. This subset tends to get the highlights in the descriptions.

<u>Nasty</u> Examples in which the program does particularly poorly are useful, if well chosen, to illustrate the weak points and pitfalls of the techniques used, the restrictions and constraints in the input, etc. They may point out improvements or extensions.

<u>Silly</u> Examples having very weak connection with the purpose or intention of the routines or algorithms discussed serve no useful end, except perhaps to point out that the maker of such examples did not understand the issues. For instance, one could take a box full of pins, drop them on the table, take their picture and ask SEE to work on it.

A collection of simple, favorable, and nasty examples follows. They are not in that order.

A discussion is found at the end of this section.

Stereo Scenes Analysis of stereographic pictures will be found in the section 'Stereo Perception'.

Finding the background Examples where the background is not known in advance and has to be deduced are given in the section 'Background Discrimination by Computer'.

LIST OF SCENES ANALYZED BY SEE IN THIS SECTION

	PAGE		
Name.	Comments.	Scene (figure).	Computer Results.
R17	107	108	109
L3	110	111	112
R3	113	114	115
SPREAD	116	117	118
STACK	119	120	122
STACK*	119	121	122
L10	123	124	125
R10	126	127	128
TOWER	129	130	131
REWOT	132	133	134
WRIST*	135	136	137
L2	138	141	142
R2	138	139	140
L19	143	144	145
R19	146	147	148
CORN	149	150	151
L9	152	153	154, 155
R9	156	158	157
R9T	156	159	160
TRIAL	161	162	163
ARCH	164	165	166
HARD	167	168	169
L4	170	171	172
R4	173	174	175
MOMO	176	177	178
BRIDGE	179	180	181

<u>Scene R17</u> The three prisms are found. In scenes like this, the position of one or two vertices may alter the analysis made by SEE, by changing radically the slope-direction of a small segment (such as KL and GH, figure 'R17'), killing several T-joints and separating regions :1-2 from :5-6.

Small errors in the coordinates of vertices K, L, G, H, and few others will drastically change the slope of segments of short length. This will transform G and K to be Arrows or Forks, so that G and K will no longer be matching T's (cf. also 'Conservatism and Tolerance' page 173). As a consequence, body :2-1 will be disconnected from body :5-6. This annoying problem is not difficult to correct, at preprocessor level, since there is good information about the slope of the (long) line BN : the slope of KL has to agree with the slope of BN, giving a good estimate of its true shape. The rule seems to be that these short segments should be "re-oriented" if necessary, to agree with the longer ones, which are more reliable. Deeper analysis is found in section 'On Noisy Input'.

SUGGESTION

The preprocessor should consider the hypothesis that BKLN are colinear -- or SEE should propose it for confirmation (see 'Division of Work in Computer Vision', p. 60).

SUGGESTION

<u>The % signs</u> In the printouts of some scenes, such as R17 (see 'RESULTS FOR R17' in page 109), a % sign appears as part of the name of every region and vertex; that is, %:3 instead of :3. This will be the case in all scenes having names starting with the letter R, differentiating the "right regions" from the "left regions". This will become clear in the section 'Stereo Perception', page 233 ; until then, disregard the %'s.

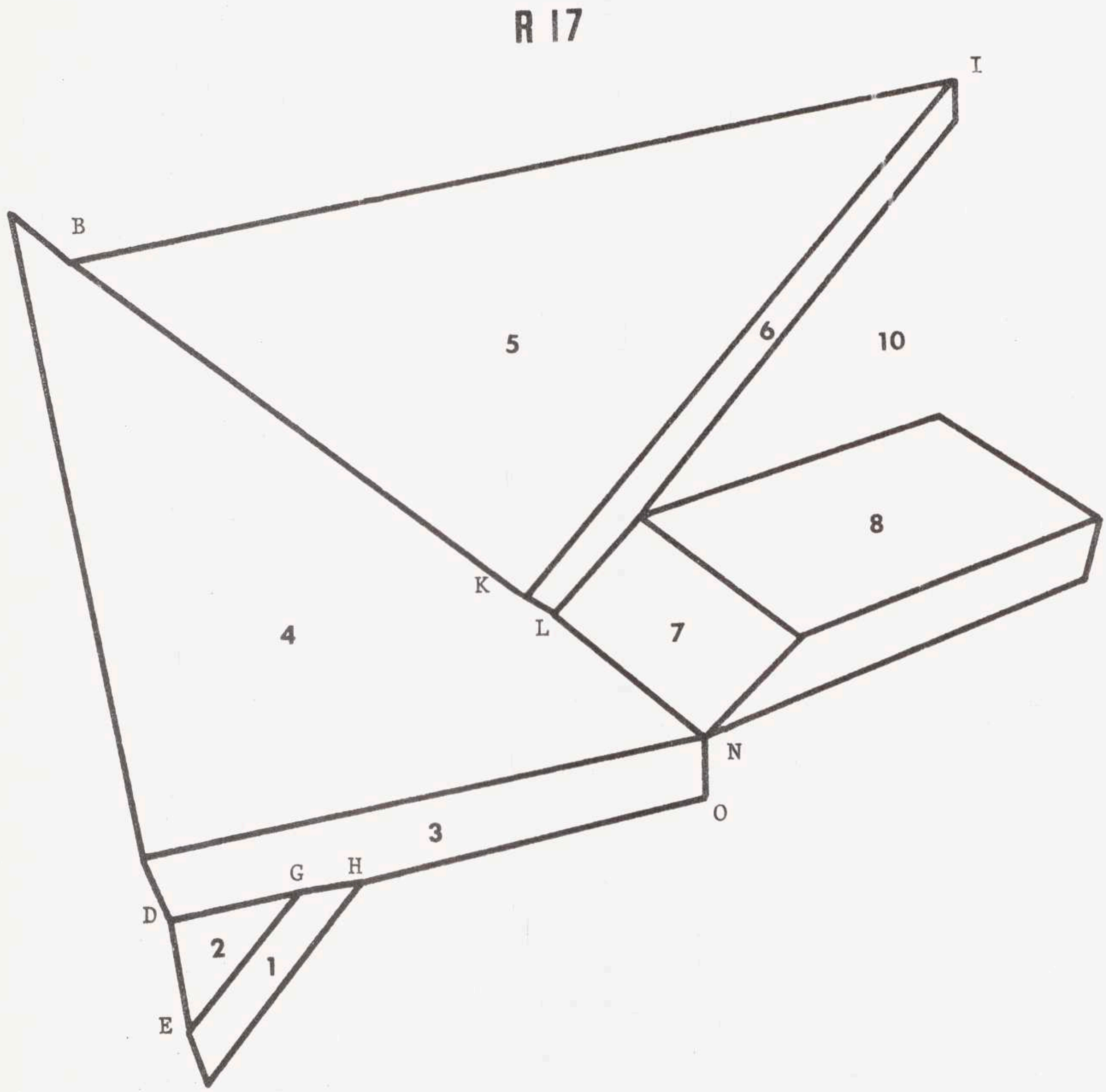


FIGURE 'R 1 7'

The three prisms were correctly found. There are several "nasty" coincidences in this scene, simulating the data that a not-too-satisfactory preprocessor will tend to provide.

```
SEE 58 ANALYZES R17
EVIDENCE
LOCALEVIDENCE
TRIANG
GLOBAL
((NIL) ((X:9) G0012 G0011 G0009) ((X:8) G0012 G0010 G0009) ((X:7) G0011 G0010) ((X:6) G0015 G0014 G0013) ((X:1) G0015 G0
013) ((X:2) G0016) ((X:3) G0017) ((X:5) G0016 G0014) ((X:4) G0017) ((X:10)))
((NIL) (NIL) (NIL) ((X:9 X:8 X:7) G0012 G0009 G0011 G0010) (NIL) ((X:6 X:1) G0014 G0015 G0013) ((X:2) G0016) ((X:3) G001
7) ((X:5) G0016 G0014) ((X:4) G0017) ((X:10)))
LOCAL
(LOCAL ASSUMES (X:6 X:1) (X:5) SAME BODY)
((NIL) (NIL) ((X:9 X:8 X:7) G0012 G0009 G0011 G0010) ((X:5 X:6 X:1) G0016 G0014 G0015 G0013) ((X:2) G0016) ((X:3) G0017)
 (NIL) ((X:4) G0017) ((X:10)))
LOCAL
(SINGLEBODY ASSUMES (X:3) (X:4) SAME BODY)
(SINGLEBODY ASSUMES (X:5 X:6 X:1) (X:2) SAME BODY)
((NIL) ((X:3 X:4) G0017) (NIL) ((X:5 X:6 X:1 X:2) G0016 G0014 G0015 G0013) ((X:9 X:8 X:7) G0012 G0009 G0011 G0010))
LOCAL
SMB
RESULTS
(BODY 1. IS X:3 X:4)
(BODY 2. IS X:5 X:6 X:1 X:2)
(BODY 3. IS X:9 X:8 X:7)
NIL
```

RESULTS FOR R 1 7

Scene L3 Without difficulty, two bodies are found. Each region contains four strong links relating it with other regions (see 'RESULTS FOR L3'). LOCAL is not needed to form nuclei; neither SINGLEBODY or SMB.

Explanation of the printout produced by the program In page 112, a printout of the results appears. The format is the same for every scene. It starts by saying

SEE 58 ANALYZES L3

which identifies the name of the program (SEE), its number (version number 58), and the scene to be analyzed (L3).

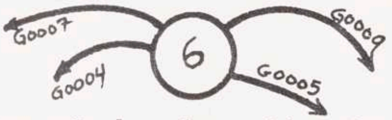
```
EVIDENCE
LOCALEVIDENCE
TRIANG
GLOBAL
```

The different sections of the program print their name, when they are entered.

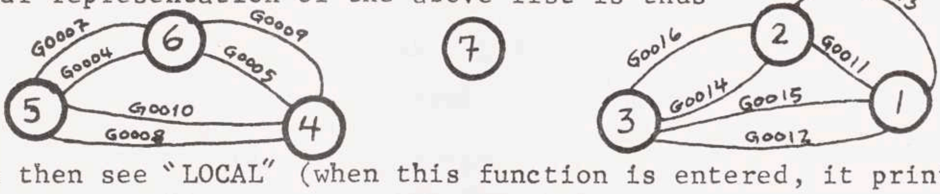
We then come to a list containing regions (such as :6) and 'gensyms' (such as G0009):

```
((NIL) ((:6) G0009 G0007 G0005 G0004) ((:5) G0010 G0008
 G0007 G0004) ((:4) G0010 G0009 G0008 G0005) ((:1) G0015
 G0013 G0012 G0011) ((:2) G0016 G0014 G0013 G0011)
 ((:3) G0016 G0015 G0014 G0012) ((:7)))
```

This list contains the nuclei and the links (strong links); the first nucleus that we see is ((:6) G0009 G0007 G0005 G0004), meaning that from nucleus (or region) :6 emanate four links, namely G0009, G0007, G0005 and G0004. We can represent this graphically:



The total representation of the above list is thus



We then see "LOCAL" (when this function is entered, it prints its name), then the list of nuclei again, this time shrunk somewhat by LOCAL (~~and GLOBAL~~); finally, we see "RESULTS", and then 2 bodies, followed by NIL, meaning the end of the program. (See page 112).

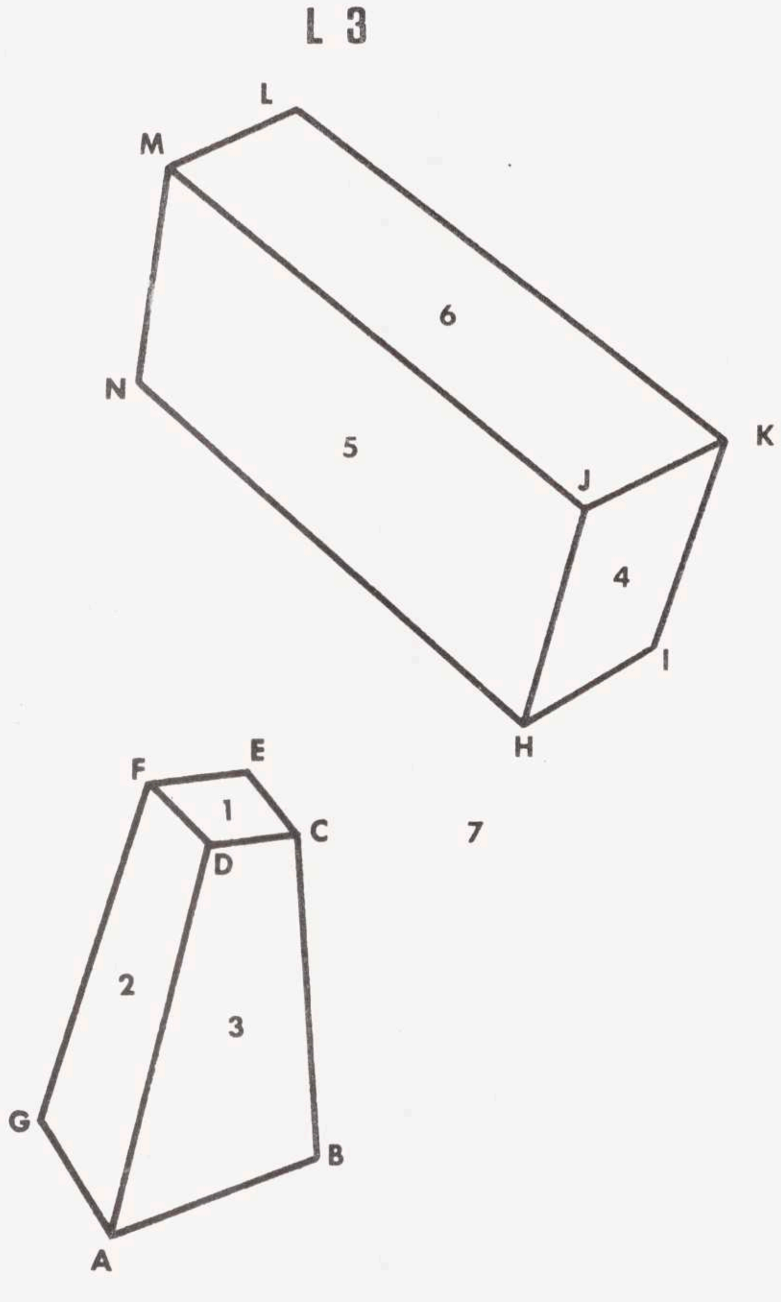


FIGURE 'L 3'

Two bodies are found in this scene by our programs.

In the input data it is indicated the fact that region :7 is the background.

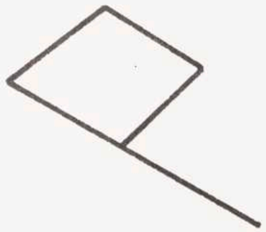
```
SEE 58 ANALYZES L3
EVIDENCE
LOCALEVIDENCE
TRIANG
GLOBAL
((NIL) ((:6) G0009 G0007 G0005 G0004) ((:5) G0010 G0008 G0007 G0004) ((:4) G0010 G0009 G0008 G0005) ((:1) G0015 G0013 G0
012 G0011) ((:2) G0016 G0014 G0013 G0011) ((:3) G0016 G0015 G0014 G0012) ((:7)))
((NIL) (NIL) (NIL) ((:6 :5 :4) G0005 G0008 G0007 G0004 G0010 G0009 G0008 G0005) (NIL) (NIL) ((:1 :2 :3) G0012 G0014 G001
3 G0011 G0016 G0015 G0014 G0012) ((:7)))
LOCAL
((NIL) (NIL) ((:6 :5 :4) G0005 G0008 G0007 G0004 G0010 G0009 G0008 G0005) (NIL) ((:1 :2 :3) G0012 G0014 G0013 G0011 G001
6 G0015 G0014 G0012) ((:7)))
LOCAL
(((:1 :2 :3) G0012 G0014 G0013 G0011 G0016 G0015 G0014 G0012) ((:6 :5 :4) G0005 G0008 G0007 G0004 G0010 G0009 G0008 G000
5))
LOCAL
SMB
RESULTS
(BODY 1. IS :1 :2 :3)
(BODY 2. IS :6 :5 :4)
NIL
```

RESULTS FOR L 3

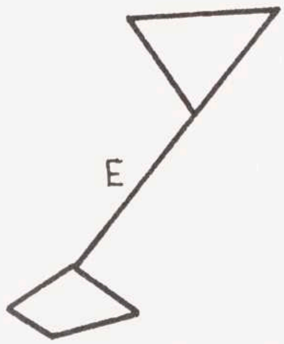
Scene R3 Two bodies are found in this scene. Vertex F is classified as of type 'T', hence only one link there exists between :2 and :4.

All scenes have regions, vertices and lines (edges) joining vertices and separating regions. We generally omit the names of the vertices from the drawing (figure 'R3'); we are also omiting the coordinate axes.

Since each region has an inside and an outside, the following are invalid or illegal configurations in a scene:



A line ending nowhere: illegal.



Our scenes should be such that, to disconnect a separate component of the graph into two components, we have to remove (delete) at least two edges. The graph above is "illegal" as input to our program, since the criterion is not met: removing edge E will disconnect the graph (cf. page 39).

Incidentally, some optical illusions are "recognized" or rejected because they come from illegal scenes of the type just described (cf. section 'Optical Illusions').

See 'Illegal scenes', page 217, in section 'On noisy input.'

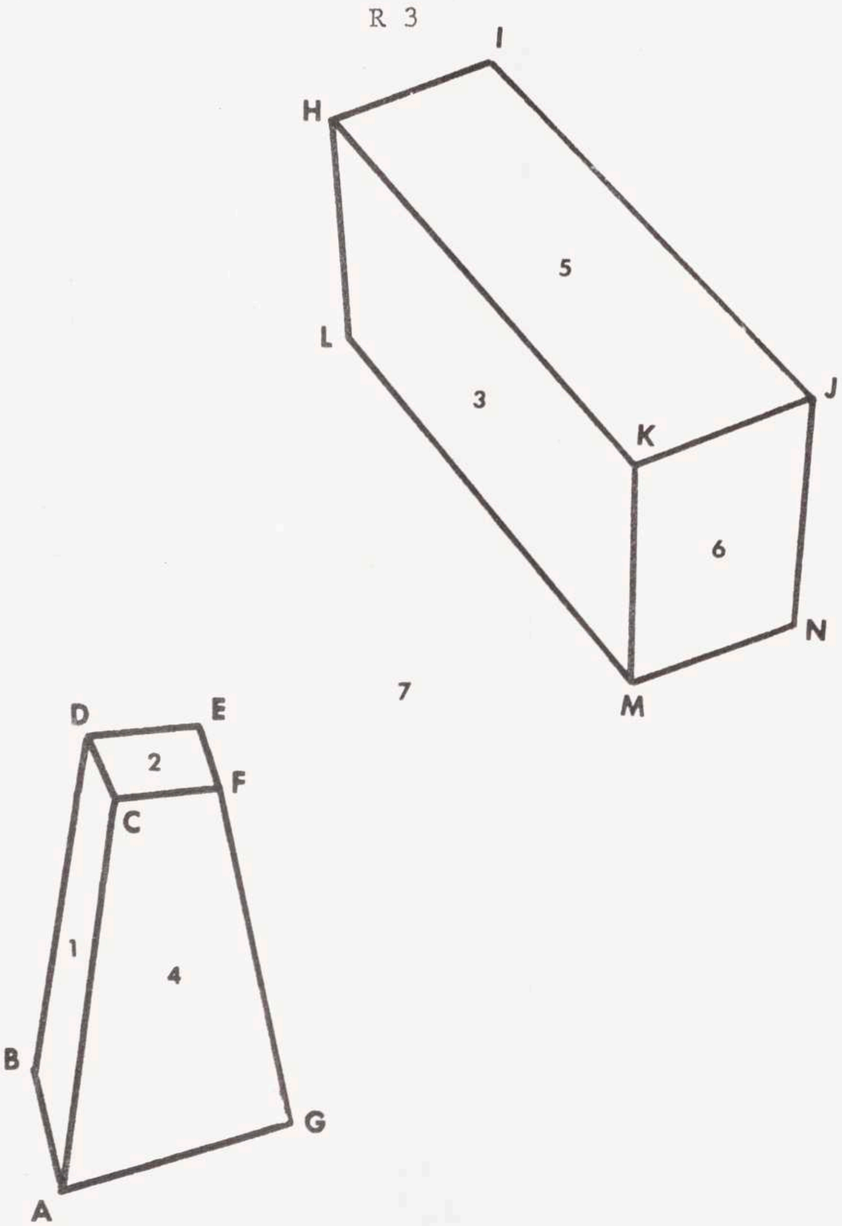


FIGURE 'R 3'

A scene analyzed by the program.

```
SEE 58 ANALYZES R3
EVIDENCE
LOCALEVIDENCE
TRIANG
GLOBAL
((NIL) ((X:6) G0009 G0008 G0006 G0005) ((X:5) G0010 G0009 G0007 G0006) ((X:3) G0010 G0008 G0007 G0005) ((X:2) G0013 G001
2 G0011) ((X:1) G0015 G0014 G0013 G0011) ((X:4) G0015 G0014 G0012) ((X:7)))
((NIL) (NIL) (NIL) ((X:6 X:5 X:3) G0005 G0009 G0006 G0010 G0008 G0007 G0005) (NIL) (NIL) ((X:2 X:1 X:4) G0015 G0013 G001
1 G0015 G0014 G0012) ((X:7)))
LOCAL
((NIL) (NIL) ((X:6 X:5 X:3) G0005 G0009 G0006 G0010 G0008 G0007 G0005) (NIL) ((X:2 X:1 X:4) G0015 G0013 G0011 G0015 G001
4 G0012) ((X:7)))
LOCAL
(((X:2 X:1 X:4) G0015 G0013 G0011 G0015 G0014 G0012) ((X:6 X:5 X:3) G0005 G0009 G0006 G0010 G0008 G0007 G0005))
LOCAL
SMB
RESULTS
(BODY 1. IS X:2 X:1 X:4)
(BODY 2. IS X:6 X:5 X:3)
NIL
```

RESULTS FOR R 3

Scene SPREAD Body :41-42 was found; also :8-18-19. In the first case, there was one strong link between :41 and :42, because of the heuristic (g) of table 'GLOBAL EVIDENCE' (page 87), and SINGLEBODY completed the object. In the second case, heuristic (g) could not be applied, and SMB had to join :19 with :18.

Bodies :29-30-31-32 and :25-26-27-28 are adequately found. Also the badly occluded long body :10-9-11-12-3 is found.

Body :21-6-25-20 is found as one body. An older version of SEE {Guzmán FJCC 68} used to report two: :6-21 and :5-20. The change is as follows: one link is placed between :6 and :5 because of the matching T's, the other link is a weak one placed because :5 and :20 form a LEG; a weak link is also placed between :6 and :5.

:24 gets reported isolated, instead of together with :22-23, because no Leg is seen; but see comment (page 30) in section 'Simplified View of Scene Analysis'.

SEE tries to find a "minimal" answer; minimal in the sense that it will try to explain the scene with the minimum possible number of bodies (cf. section 'The Concept of a Body'). That is the reason which joined :41 and :42 in one body, instead of two, which is other possible correct answer. That is also true of :19-18-8, interpreted as one parallelepiped with a vertical face (:19) and an horizontal face (:18-8).

The background of SPREAD is also computed (see page 226 of section 'Background Discrimination by Computer').

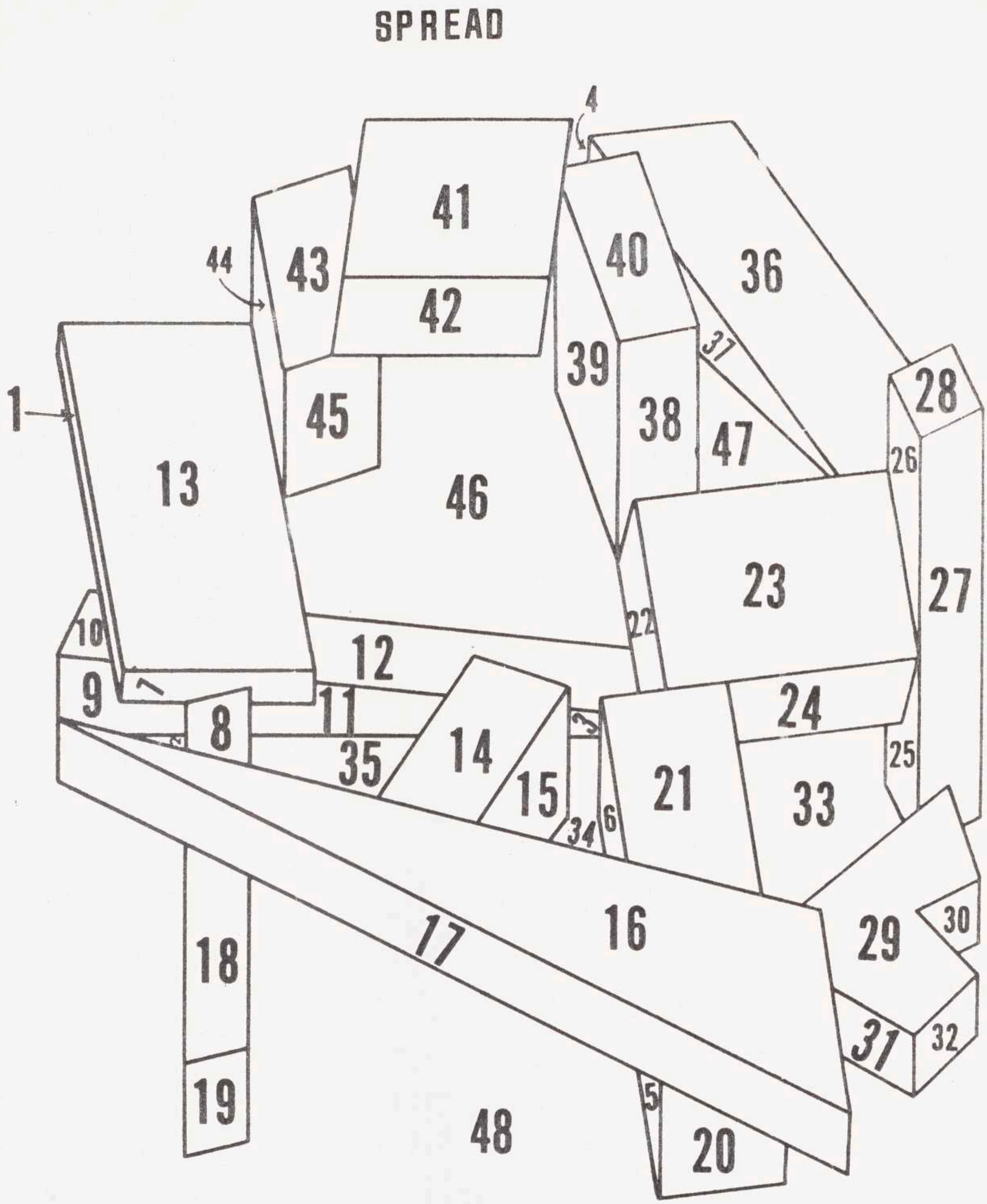


FIGURE 'S P R E A D'

Bodies :10-9-11-12-3 and :6-21-5-20 are properly found. Also is correctly identified the body :19-18-8, which is a parallelepiped with a vertical face (:19) and an horizontal face (:8-18).

```
(QUOTE (SINTO = 0.05, COLTO = 0.01))

SEE 58 ANALYZES SPREAD
EVIDENCE
LOCALEVIDENCE
TRIANG
GLOBAL
((NIL) ((*15) G0028 G0025) ((*14) G0028 G0025) ((*35)) ((*19)) ((*18) G0031 G0029) ((*31) G0037 G0035 G0034) ((*32) G003
8 G0037 G0036 G0035) ((*34)) ((*28) G0047 G0046 G0044 G0043) ((*26) G0047 G0045 G0044 G0042) ((*45) G0053 G0052 G0050) (
(*42) G0049) ((*43) G0054 G0053 G0051 G0050) ((*44) G0054 G0052 G0051) ((*16) G0033) ((*2)) ((*8) G0031 G0029) ((*21) G0
061 G0040) ((*6) G0061 G0041 G0040) ((*3) G0027 G0026) ((*13) G0062 G0058 G0056 G0055) ((*12) G0065 G0063) ((*11) G0064
G0060 G0027 G0026) ((*7) G0062 G0059 G0058 G0057) ((*1) G0059 G0057 G0056 G0055) ((*9) G0064 G0060 G0030) ((*10) G0065 G
0063 G0030) ((*5) G0041 G0032) ((*17) G0033) ((*20) G0032) ((*30) G0039) ((*27) G0066 G0046 G0045 G0043) ((*29) G0039 G0
038 G0036 G0034) ((*24)) ((*25) G0066) ((*33)) ((*37) G0067) ((*47)) ((*36) G0048) ((*4) G0067 G0048) ((*48)) ((*40) G00
23 G0022 G0021) ((*41) G0049) ((*23) G0068 G0042) ((*39) G0024 G0023) ((*38) G0024 G0022 G0021) ((*22) G0068) ((*46)))
((NIL) (NIL) ((*15 *14) G0025 G0028 G0025) ((*35)) ((*19)) (NIL) (NIL) (NIL) ((*34)) (NIL) (NIL) (NIL) ((*42) G0049) (NI
L) ((*45 *43 *44) G0054 G0053 G0050 G0054 G0052 G0051) ((*16) G0033) ((*2)) ((*18 *8) G0029 G0031 G0029) (NIL) ((*21 *6)
 G0040 G0061 G0041 G0040) (NIL) (NIL) (NIL) (NIL) (NIL) ((*13 *7 *1) G0056 G0062 G0058 G0059 G0057 G0056 G0055) ((*3 *11
 *9) G0064 G0027 G0026 G0064 G0060 G0030) ((*12 *10) G0063 G0065 G0063 G0030) ((*5) G0041 G0032) ((*17) G0033) ((*20) G0
032) ((*30) G0039) ((*28 *26 *27) G0043 G0047 G0044 G0042 G0066 G0046 G0045 G0043) ((*31 *32 *29) G0034 G0037 G0035 G003
9 G0038 G0036 G0034) ((*24)) ((*25) G0066) ((*33)) ((*37) G0067) ((*47)) ((*36) G0048) ((*4) G0067 G0048) ((*48)) (NIL)
((*41) G0049) ((*23) G0068 G0042) (NIL) ((*39 *40 *38) G0023 G0023 G0024 G0022 G0021) ((*22) G0068) ((*46)))
LOCAL
(LOCAL ASSUMES (*36) (*4) SAME BODY)
(LOCAL ASSUMES (*30) (*31 *32 *29) SAME BODY)
(LOCAL ASSUMES (*16) (*17) SAME BODY)
(LOCAL ASSUMES (*21 *6) (*5) SAME BODY)
(LOCAL ASSUMES (*5 *21 *6) (*20) SAME BODY)
(LOCAL ASSUMES (*3 *11 *9) (*12 *10) SAME BODY)
LOCAL
((NIL) ((*15 *14) G0025 G0028 G0025) ((*35)) ((*19)) (NIL) ((*34)) (NIL) ((*42) G0049) ((*45 *43 *44) G0054 G0053 G0050
G0054 G0052 G0051) ((*17 *16) G0033) ((*2)) ((*18 *8) G0029 G0031 G0029) ((*20 *5 *21 *6) G0032 G0061 G0041 G0040) (NIL)
 (NIL) ((*13 *7 *1) G0056 G0062 G0058 G0059 G0057 G0056 G0055) ((*12 *10 *3 *11 *9) G0065 G0063 G0064 G0027 G0026 G0064
G0060 G0030) (NIL) (NIL) (NIL) (NIL) ((*31 *32 *29 *30) G0037 G0035 G0038 G0036 G0034 G0039) ((*28 *26 *27) G0043 G0047
G0044 G0042 G0066 G0046 G0045 G0043) (NIL) ((*24)) ((*25) G0066) ((*33)) ((*37) G0067) ((*47)) ((*4 *36) G0067 G0048) (N
IL) ((*48)) ((*41) G0049) ((*23) G0068 G0042) ((*39 *40 *38) G0023 G0023 G0024 G0022 G0021) ((*22) G0068) ((*46)))
LOCAL
(SINGLEBODY ASSUMES (*23) (*22) SAME BODY)
(SINGLEBODY ASSUMES (*42) (*41) SAME BODY)
(SINGLEBODY ASSUMES (*4 *36) (*37) SAME BODY)
(SINGLEBODY ASSUMES (*28 *26 *27) (*25) SAME BODY)
((NIL) ((*39 *40 *38) G0023 G0023 G0024 G0022 G0021) ((*23 *22) G0068 G0042) (NIL) ((*4 *36 *37) G0067 G0048) (NIL) (NIL
) ((*24)) ((*28 *26 *27 *25) G0043 G0047 G0044 G0042 G0066 G0046 G0045 G0043) ((*31 *32 *29 *30) G0037 G0035 G0038 G0036
 G0034 G0039) ((*12 *10 *3 *11 *9) G0065 G0063 G0064 G0027 G0026 G0064 G0060 G0030) ((*13 *7 *1) G0056 G0062 G0058 G0059
 G0057 G0056 G0055) ((*20 *5 *21 *6) G0032 G0061 G0041 G0040) ((*18 *8) G0029 G0031 G0029) ((*17 *16) G0033) ((*45 *43 *
44) G0054 G0053 G0050 G0054 G0052 G0051) ((*42 *41) G0049) ((*19)) ((*15 *14) G0025 G0028 G0025))
LOCAL
SMB
(SMB ASSUMES *19 *18 SAME BODY)
RESULTS
(THE FIRST 1. BODIES ARE ((*24)))
(BODY 2. IS *39 *40 *38)
(BODY 3. IS *23 *22)
(BODY 4. IS *4 *36 *37)
(BODY 5. IS *28 *26 *27 *25)
(BODY 6. IS *31 *32 *29 *30)
(BODY 7. IS *12 *10 *3 *11 *9)
(BODY 8. IS *13 *7 *1)
(BODY 9. IS *20 *5 *21 *6)
(BODY 10. IS *17 *16)
(BODY 11. IS *45 *43 *44)
(BODY 12. IS *42 *41)
(BODY 13. IS *19 *18 *8)
(BODY 14. IS *15 *14)
NIL
```

RESULTS FOR S P R E A D

Scenes STACK and STACK* In both cases all the bodies were accurately identified by our program, which is written in LISP. In both cases the body :4-15-16 is isolated.

These scenes show that in many instances one could drastically alter the position of a vertex, without modifying the output of SEE (compare figure 'STACK' with 'STACK*').

Other examples would show that the vertices of type 'L' can be arbitrarily displaced, so long as their type remains 'L' and other vertices do not change type, without detrimental effect. This displacement may possibly affect some heuristics that use concepts of parallelism or colinearity, but not the rules that use the shape or type of a vertex (cf. table 'VERTICES', page 69) for placing and inhibiting links. Read 'Misplaced vertices' in page 211 , in section 'On noisy input.'

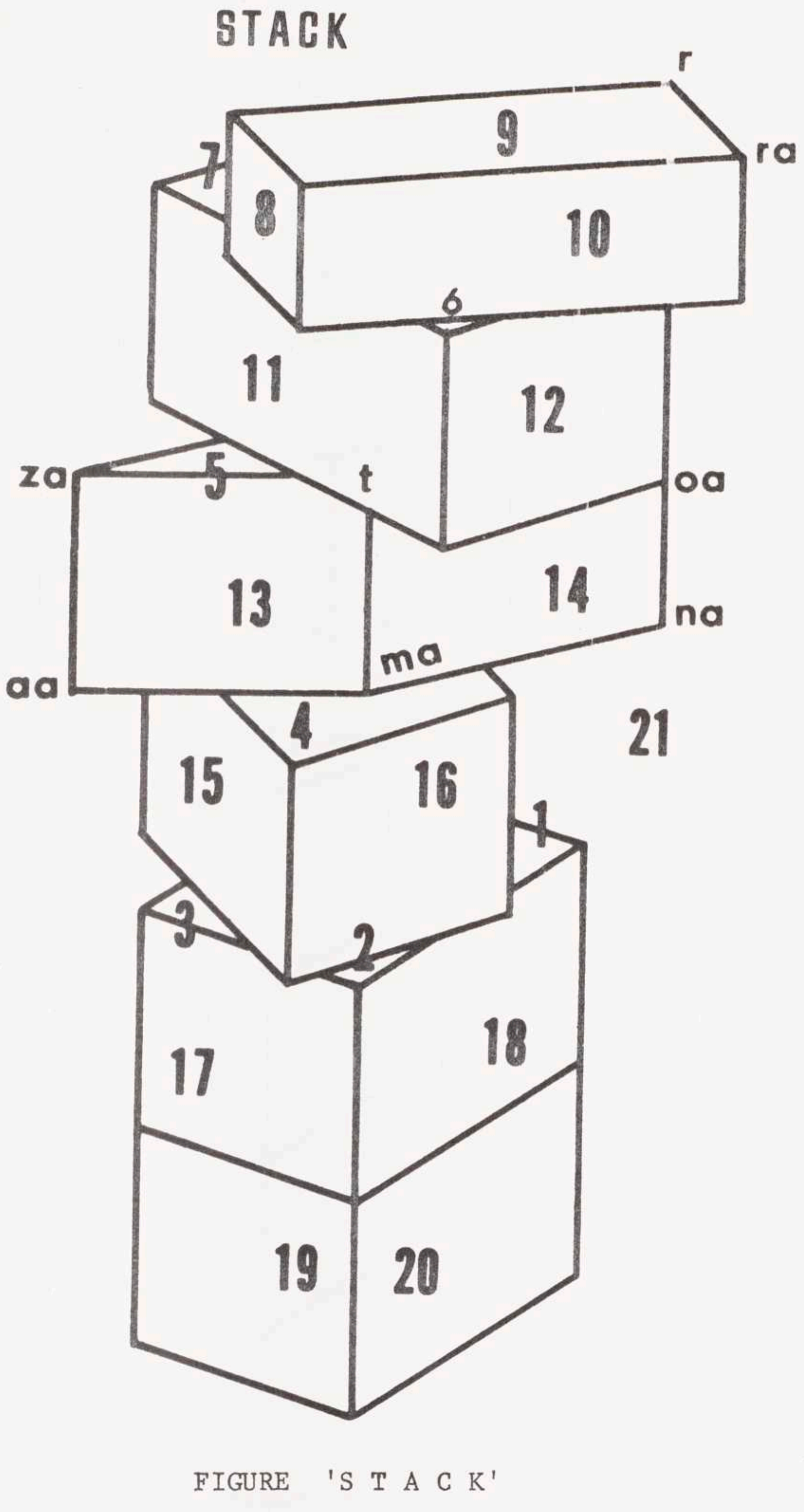


FIGURE 'S T A C K'

Every body is correctly identified. Compare with scene STACK*. This pair of drawings illustrate the fact that it is often possible to disturb the coordinates (the position) of a vertex, without introducing errors in the recognition.

STACK *

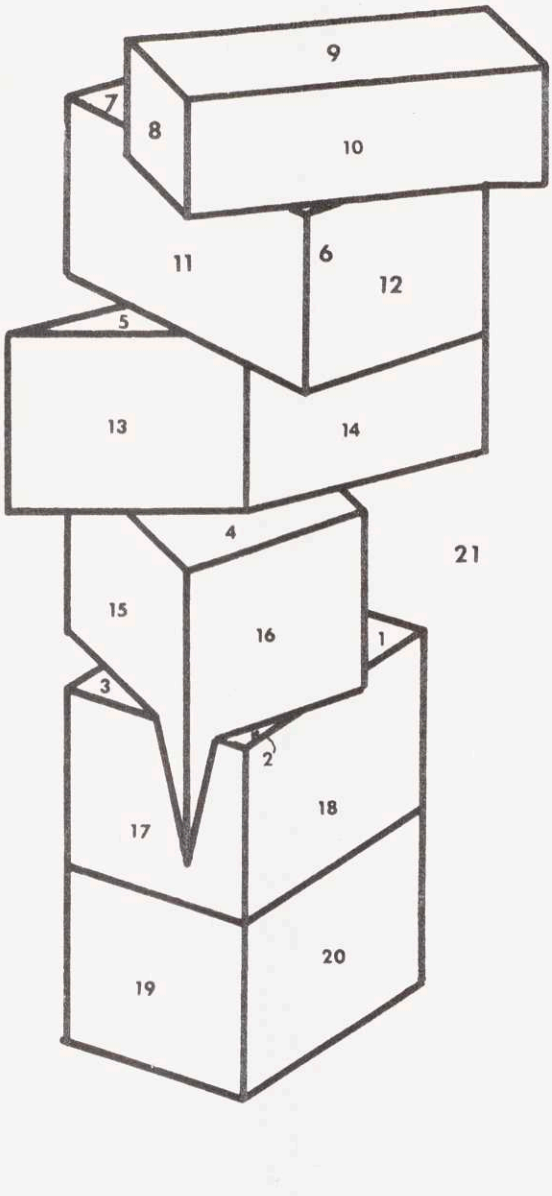


FIGURE 'S T A C K *'

Every body is correctly found. Compare with scene STACK.

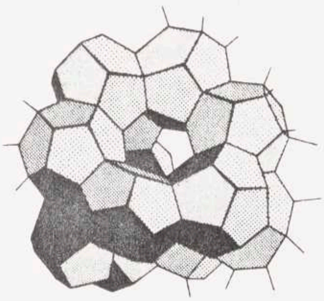
```
SEE 58 ANALYZES II (STACK , STACK*)
EVIDENCE
LOCALEVIDENCE
TRIANG
GLOBAL
((NIL) ((:20) G0047 G0046) ((:19) G0047 G0046) ((:5) G0036) ((:13) G0051 G0040 G0036) ((:9) G0056 G0054 G0053 G0039) ((:
7) G0057 G0052) ((:8) G0056 G0055 G0054 G0038) ((:11) G0060 G0059 G0052 G0050) ((:6) G0059 G0058 G0057) ((:10) G0055 G00
53 G0039 G0038) ((:12) G0060 G0058 G0050) ((:4) G0061 G0049 G0042 G0041) ((:14) G0051 G0040) ((:21), ((:1) G0062 G0044)
((:18) G0065 G0063 G0045 G0044) ((:16) G0061 G0043 G0041 G0037) ((:2) G0066 G0064 G0063 G0062) ((:3) G0066 G0048) ((:15)
 G0049 G0043 G0042 G0037) ((:17) G0065 G0064 G0048 G0045))
((NIL) (NIL) ((:20 :19) G0046 G0047 G0046) ((:5) G0036) (NIL) (NIL) ((:7) G0057 G0052) (NIL) (NIL) (NIL) ((:9 :8 :10) G0
053 G0056 G0054 G0055 G0053 G0039 G0038) ((:6 :11 :12) G0058 G0057 G0059 G0052 G0060 G0058 G0050) (NIL) ((:13 :14) G0040
 G0036 G0051 G0040) ((:21)) ((:1) G0062 G0044) (NIL) (NIL) (NIL) (NIL) ((:4 :16 :15) G0042 G0061 G0041 G0049 G0043 G0042
 G0037) ((:3 :2 :18 :17) G0048 G0066 G0062 G0063 G0044 G0065 G0064 G0046 G0045))
((NIL) ((:20 :19) G0046 G0047 G0046) ((:5) G0036) (NIL) (NIL) (NIL) ((:9 :8 :10) G0053 G0056 G0054 G0055 G0053 G0039 G00
38) ((:7 :6 :11 :12) G0052 G0057 G0059 G0052 G0060 G0058 G0050) ((:13 :14) G0040 G0036 G0051 G0040) ((:21)) (NIL) (NIL)
(NIL) ((:4 :16 :15) G0042 G0061 G0041 G0049 G0043 G0042 G0037) ((:1 :3 :2 :18 :17) G0044 G0066 G0062 G0063 G0044 G0065 G
0064 G0048 G0045))
LOCAL
(LOCAL ASSUMES (:5) (:13 :14) SAME BODY)
LOCAL
((NIL) ((:20 :19) G0046 G0047 G0046) ((:13 :14 :5) G0036 G0051 G0040 G0036) (NIL) ((:9 :8 :10) G0053 G0056 G0054 G0055 G
0053 G0039 G0038) ((:7 :6 :11 :12) G0052 G0057 G0059 G0052 G0060 G0058 G0050) (NIL) ((:21)) (NIL) ((:4 :16 :15) G0042 G0
061 G0041 G0049 G0043 G0042 G0037) ((:1 :3 :2 :18 :17) G0044 G0066 G0062 G0063 G0044 G0065 G0064 G0048 G0045))
LOCAL
(((:1 :3 :2 :18 :17) G0044 G0066 G0062 G0063 G0044 G0065 G0064 G0048 G0045) ((:4 :16 :15) G0042 G0061 G0041 G0049 G0043
G0042 G0037) ((:7 :6 :11 :12) G0052 G0057 G0059 G0052 G0060 G0058 G0050) ((:9 :8 :10) G0053 G0056 G0054 G0055 G0053 G003
9 G0038) ((:13 :14 :5) G0036 G0051 G0040 G0036) ((:20 :19) G0046 G0047 G0046))
LOCAL
SMB
RESULTS
(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)
NIL
```

RESULTS FOR STACK AND STACK*

Scene L10 The concave object :11-15-14-7-6 presents no problem, since there are plenty of visible vertices (figure 'L10'), and SEE makes good use of them.

SINGLEBODY is necessary to join regions :13 and :2.

The bodies of a scene do not need to be prismatic in shape, nor convex. Their vertices could have errors in their two-dimensional position. Table 'ASSUMPTIONS' specifies the suppositions that our program obeys.



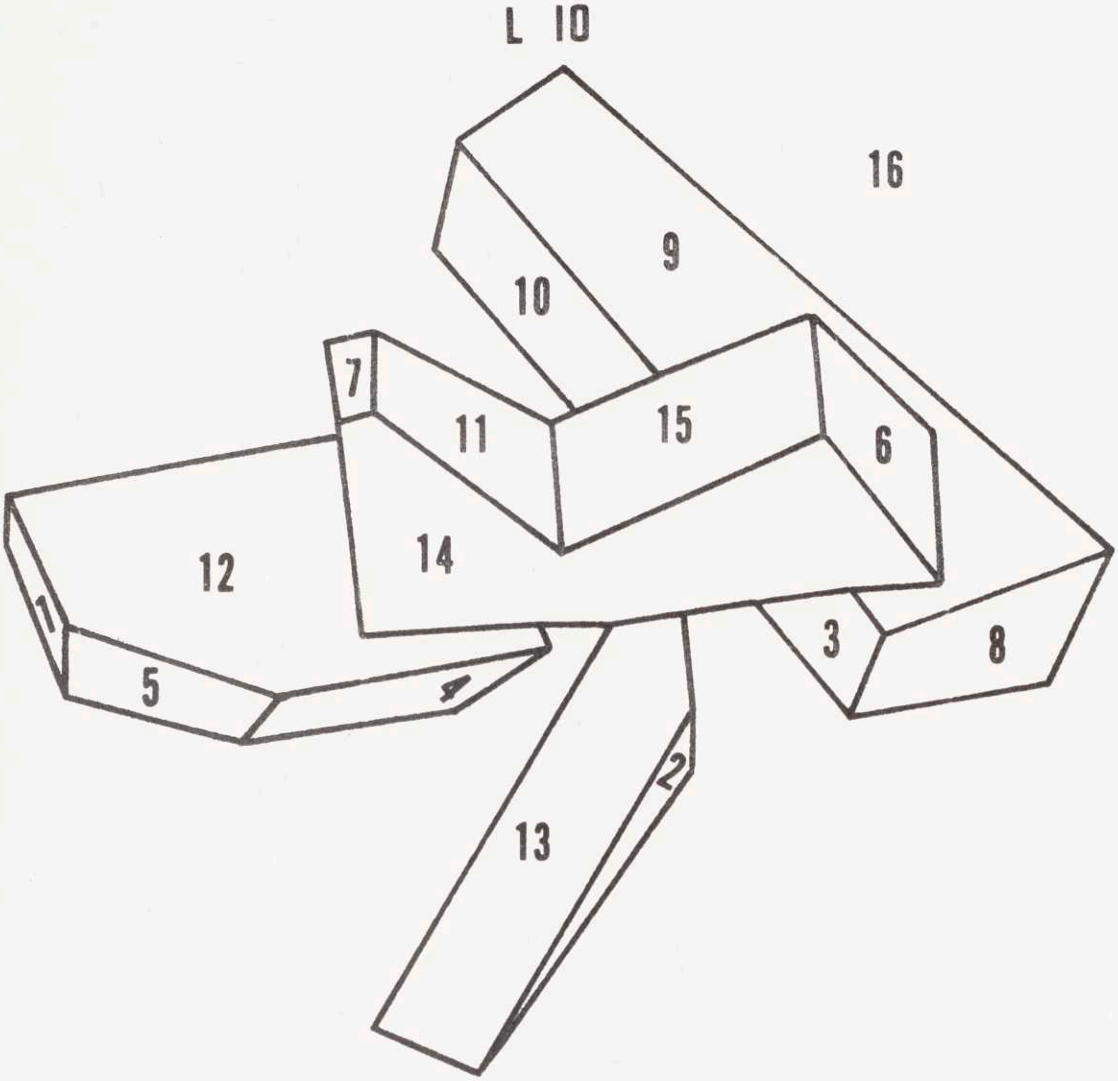


FIGURE 'L 1 0'

Singlebody had to join :2 with :13.
All four bodies were happily identified.

```
SEE 58 ANALYZES L10
EVIDENCE
LOCALEVIDENCE
TRIANG
GLOBAL
((NIL) ((:5) G0040 G0039 G0037 G0035 G0034 G0033) ((:1) G0041 G0039 G0038 G0037) ((:8) G0045 G0044 G0042 G0032) ((:6) G0
049 G0048 G0047 G0046) ((:9) G0051 G0050 G0045 G0043 G0032) ((:10) G0059 G0051 G0050) ((:7) G0057 G0056 G0055 G0054) ((:
2) G0058) ((:3) G0059 G0044 G0043 G0042) ((:13) G0058) ((:11) G0060 G0055 G0054 G0053) ((:15) G0060 G0052 G0049 G0048) (
(:14) G0057 G0056 G0053 G0052 G0047 G0046) ((:4) G0061 G0036 G0035 G0033) ((:12) G0061 G0041 G0040 G0038 G0036 G0034) ((
:16)))
((NIL) (NIL) (NIL) (NIL) (NIL) (NIL) (NIL) (NIL) ((:2) G0058) ((:8 :9 :10 :3) G0042 G0045 G0032 G0051 G0050 G0059 G0044
G0043 G0042) ((:13) G0058) (NIL) (NIL) ((:6 :15 :7 :11 :14) G0047 G0060 G0049 G0048 G0060 G0055 G0054 G0057 G0056 G0053
G0052 G0047 G0046) (NIL) ((:5 :1 :4 :12) G0035 G0041 G0039 G0037 G0036 G0035 G0033 G0061 G0041 G0040 G0038 G0036 G0034)
((:16)))
LOCAL
((NIL) (NIL) (NIL) (NIL) ((:2) G0058) ((:8 :9 :10 :3) G0042 G0045 G0032 G0051 G0050 G0059 G0044 G0043 G0042) ((:13) G005
8) (NIL) ((:6 :15 :7 :11 :14) G0047 G0060 G0049 G0048 G0060 G0055 G0054 G0057 G0056 G0053 G0052 G0047 G0046) ((:5 :1 :4
:12) G0035 G0041 G0039 G0037 G0036 G0035 G0033 G0061 G0041 G0040 G0038 G0036 G0034) ((:16)))
LOCAL
(SINGLEBODY ASSUMES (:2) (:13) SAME BODY)
(((:5 :1 :4 :12) G0035 G0041 G0039 G0037 G0036 G0035 G0033 G0061 G0041 G0040 G0038 G0036 G0034) ((:6 :15 :7 :11 :14) G00
47 G0060 G0049 G0048 G0060 G0055 G0054 G0057 G0056 G0053 G0052 G0047 G0046) (NIL) ((:8 :9 :10 :3) G0042 G0045 G0032 G005
1 G0050 G0059 G0044 G0043 G0042) ((:2 :13) G0058))
LOCAL
SMB
RESULTS
(BODY 1. IS :5 :1 :4 :12)
(BODY 2. IS :6 :15 :7 :11 :14)
(BODY 3. IS :8 :9 :10 :3)
(BODY 4. IS :2 :13)
NIL

L10
```

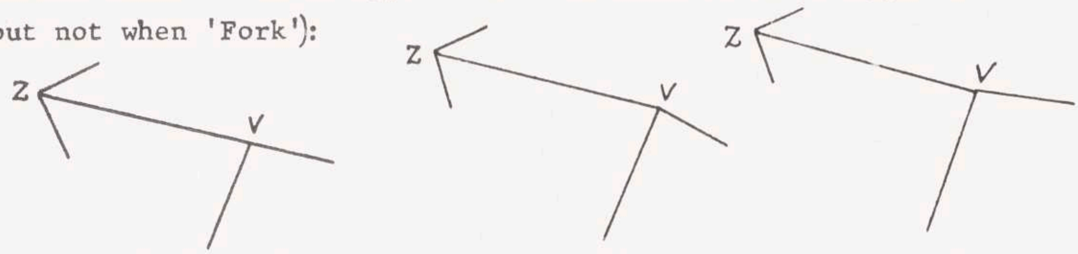
RESULTS FOR L10

Scene R10 Four bodies are found by our program in R10. The scene is a good example of a "noisy" scene, in which edges that should be straight look crooked. This is because the coordinates of each vertex are "imprecise"; the vertices have some error in their coordinates. Other scenes also show this tendency; they accurately represent the data analyzed by SEE (the scenes in their final form were drawn by program, then inked manually), and should not be considered as "sloppy drawing jobs".

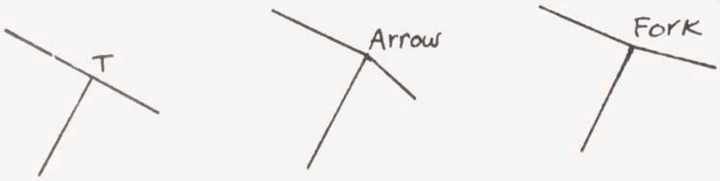
SEE has several ways to cope with these imperfections:

(1) tolerant definitions of parallelism and colinearity.

(2) insensitivity of heuristics to displacements of the vertex. For instance, vertex V will inhibit the link that Z proposes, either when V is of type 'Arrow' or when it is of type 'T' (but not when 'Fork'):



(3) Large variations in the coordinates of a vertex are possible before that vertex changes type. Vertex of type 'T' are an exception, changing into a Fork or an Arrow by a small displacement.



Nevertheless, it is possible to "straighten" these vertices, by following the suggestion in the comments to scene R17.

The section 'On Noisy Input' deals with these matters.

R 10

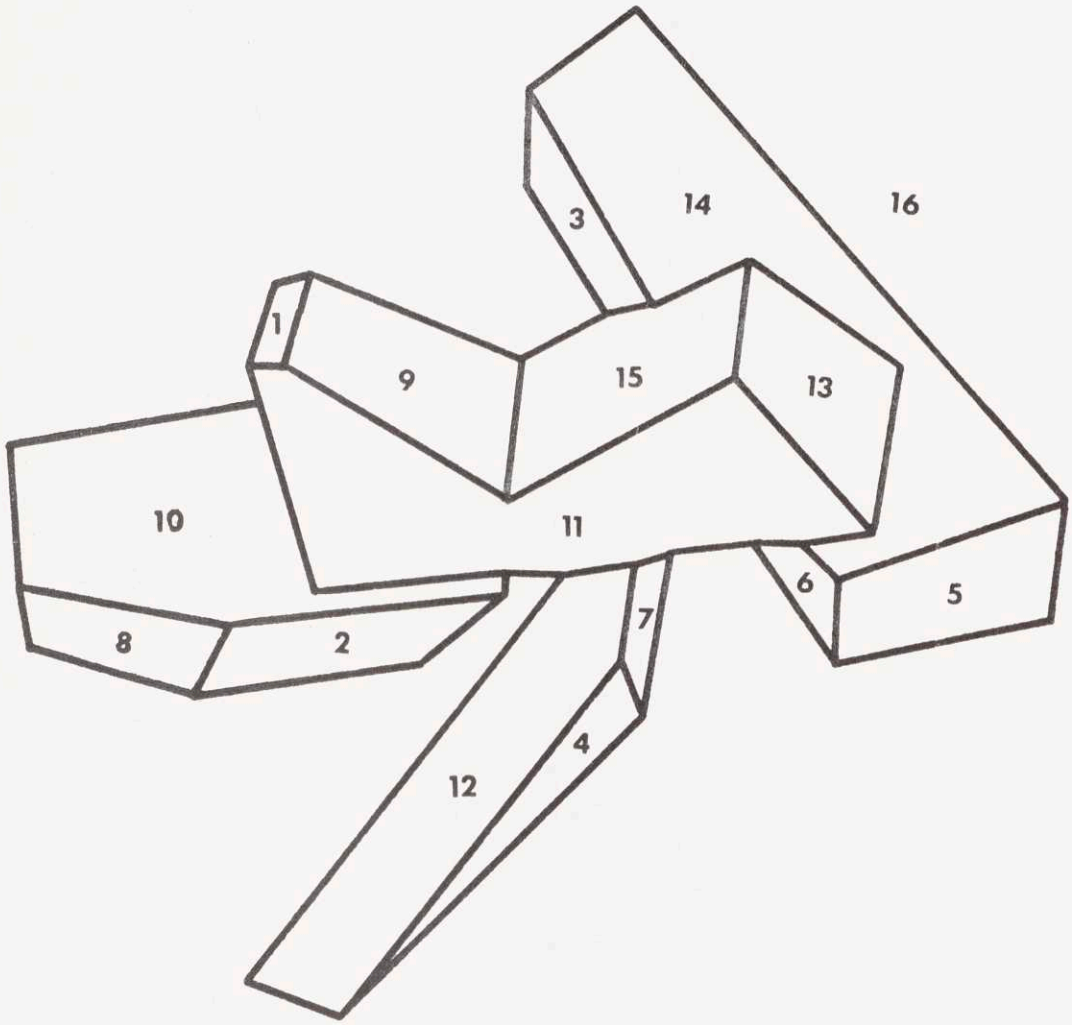


FIGURE 'R 1 0'

The scene contains "noisy" vertices; hence, some edges look bent. SEE has resources to cope with these problems.

Figures L10 and R10 form a stereo pair. In figure 'L10 - R10' in page 247, information from both scenes is combined to find the position of these objects in three-dimensional space. See section 'Stereo Perception'.

```
SEE 58 ANALYZES R10
EVIDENCE
LOCALEVIDENCE
TRIANG
GLOBAL
((NIL) ((X:8) G0018 G0017 G0016) ((X:1) G0023 G0022 G0021 G0020) ((X:4) G0026 G0024) ((X:5) G0030 G0029 G0027 G0015) ((X
:13) G0031 G0014 G0013 G0012) ((X:6) G0032 G0029 G0028 G0027) ((X:7) G0026 G0025 G0024) ((X:12) G0025) ((X:2) G0033 G001
9 G0018 G0016) ((X:10) G0033 G0019 G0017) ((X:11) G0036 G0035 G0031 G0023 G0022 G0013) ((X:9) G0036 G0034 G0021 G0020) (
(X:15) G0035 G0034 G0014 G0012) ((X:3) G0038 G0037) ((X:14) G0038 G0037 G0032 G0030 G0028 G0015) ((X:16)))
((NIL) (NIL) (NIL) (NIL) (NIL) (NIL) (NIL) ((X:4 X:7) G0024 G0026 G0025 G0024) ((X:12) G0025) (NIL) ((X:8 X:2 X:10) G003
3 G0018 G0016 G0033 G0019 G0017) (NIL) (NIL) ((X:13 X:1 X:11 X:9 X:15) G0012 G0035 G0031 G0023 G0022 G0013 G0036 G0021 G
0020 G0035 G0034 G0014 G0012) (NIL) ((X:3 X:5 X:6 X:14) G0037 G0032 G0029 G0027 G0038 G0037 G0032 G0030 G0028 G0015) ((X
:16)))
LOCAL
((NIL) (NIL) (NIL) (NIL) ((X:4 X:7) G0024 G0026 G0025 G0024) ((X:12) G0025) ((X:8 X:2 X:10) G0033 G0018 G0016 G0033 G001
9 G0017) (NIL) ((X:13 X:1 X:11 X:9 X:15) G0012 G0035 G0031 G0023 G0022 G0013 G0036 G0021 G0020 G0035 G0034 G0014 G0012)
((X:3 X:5 X:6 X:14) G0037 G0032 G0029 G0027 G0038 G0037 G0032 G0030 G0028 G0015) ((X:16)))
LOCAL
(SINGLEBODY ASSUMES (X:4 X:7) (X:12) SAME BODY)
(((X:3 X:5 X:6 X:14) G0037 G0032 G0029 G0027 G0038 G0037 G0032 G0030 G0028 G0015) ((X:13 X:1 X:11 X:9 X:15) G0012 G0035
G0031 G0023 G0022 G0013 G0036 G0021 G0020 G0035 G0034 G0014 G0012) ((X:8 X:2 X:10) G0033 G0018 G0016 G0033 G0019 G0017)
(NIL) ((X:4 X:7 X:12) G0024 G0026 G0025 G0024))
LOCAL
SMB
RESULTS
(BODY 1. IS X:3 X:5 X:6 X:14)
(BODY 2. IS X:13 X:1 X:11 X:9 X:15)
(BODY 3. IS X:8 X:2 X:10)
(BODY 4. IS X:4 X:7 X:12)
NIL
```

RESULTS FOR R10

<u>Scene TOWER</u> There is no need to make use of LOCAL or SINGLEBODY in this scene, since there are plenty of global (strong) links among the different regions. :18-22 and :17-23 get links thanks to the heuristic that analyzes vertex of type "X".

There are several "false" vertices, formed by coindicences of edges and "genuine" vertices: the vertex common to :9, 11, 12 and 13; the one common to :2, 4, 5, 6. They do not cause problem, because

(1) in the case of the vertex common to :9, 11, 12 and 13, it is of type "MULTI', and no link is laid.

(2) In the case of the vertex shared by regions :2, 4, 5, and 6, it is an "X" that will establish one link between :4 and :5 (which is correct), and another between :2 and :6 (which will do <u>no</u> harm, since we need two "wrong" or misplaced links to cause a recognition mistake).

Compare with scene 'REWOT'.

TOWER

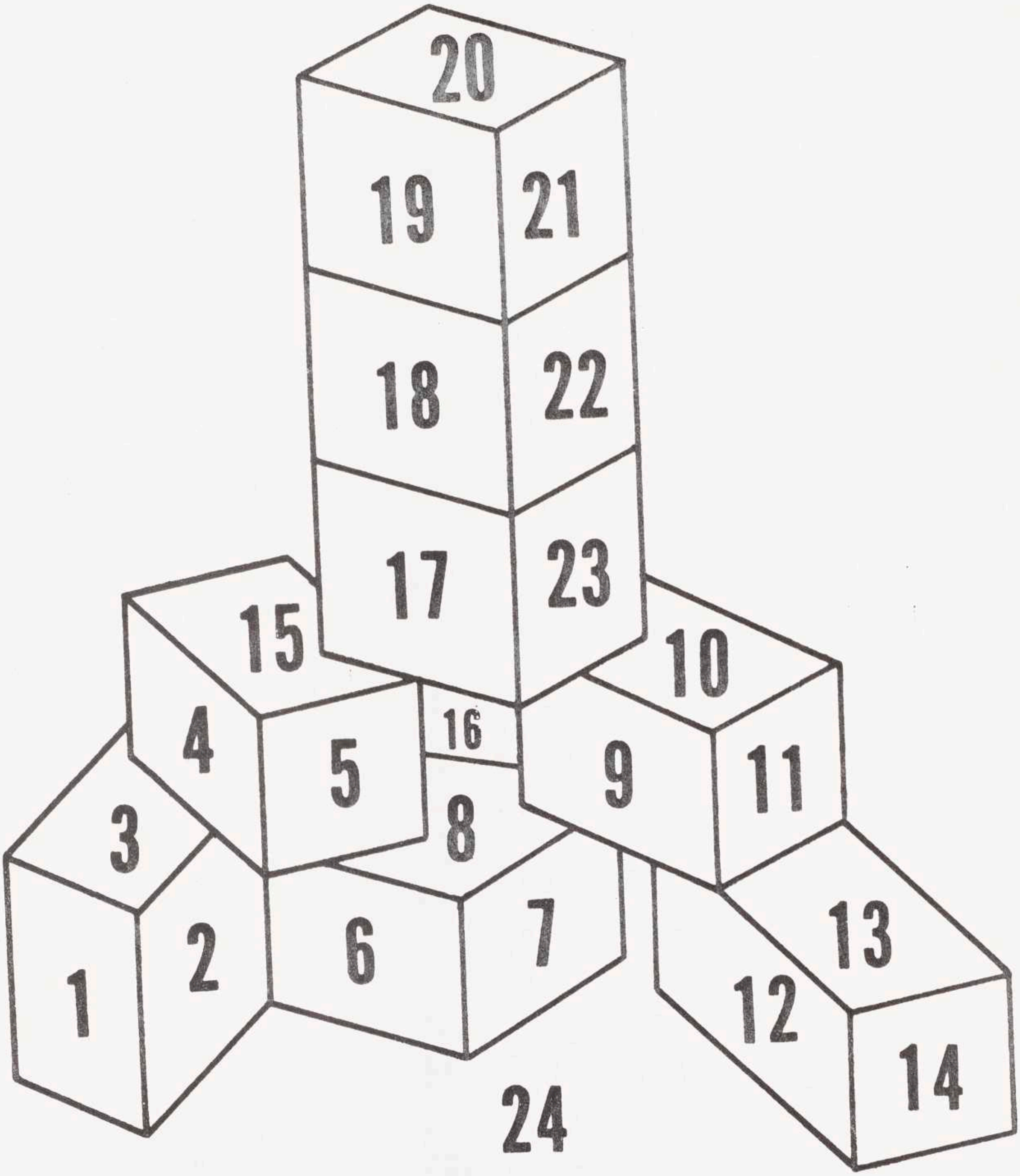


FIGURE 'T O W E R'

A "wrong" link is placed between :2 and :6, without serious consequences. Results for this scene are in "RESULTS FOR TOWER'.

```
SEE 58 ANALYZES TOWER
EVIDENCE
LOCALEVIDENCE
TRIANG
GLOBAL
((NIL) ((:20) G0019 G0018 G0017 G0016) ((:19) G0022 G0020 G0019 G0016) ((:21) G0022 G0020 G0018 G0017) ((:18) G0023 G002
1) ((:22) G0023 G0021) ((:23) G0026 G0024) ((:10) G0041 G0029 G0028 G0025) ((:11) G0041 G0028 G0027) ((:13) G0042 G0032
G0030) ((:14) G0043 G0042 G0032 G0031) ((:6) G0046 G0044 G0036 G0035 G0033) ((:2) G0047 G0046 G0040 G0038 G0037) ((:12)
G0043 G0031 G0030) ((:7) G0036 G0034 G0033) ((:9) G0029 G0027 G0025) ((:8) G0044 G0035 G0034) ((:5) G0045 G0014 G0013) (
(:16)) ((:17) G0026 G0024) ((:15) G0048 G0015 G0014) ((:4) G0048 G0045 G0015 G0013) ((:3) G0049 G0047 G0039 G0038) ((:24
)) ((:1) G0049 G0040 G0039 G0037))
((NIL) (NIL) (NIL) ((:20 :19 :21) G0017 G0020 G0019 G0016 G0022 G0020 G0018 G0017) (NIL) ((:18 :22) G0021 G0023 G0021) (
NIL) (NIL) (NIL) (NIL) (NIL) (NIL) (NIL) ((:13 :14 :12) G0030 G0042 G0032 G0043 G0031 G0030) (NIL) ((:10 :11 :9) G0025 G
0041 G0028 G0029 G0027 G0025) ((:6 :7 :8) G0046 G0035 G0036 G0033 G0044 G0035 G0034) (NIL) ((:16)) ((:23 :17) G0024 G002
6 G0024) (NIL) ((:15 :5 :4) G0015 G0014 G0048 G0045 G0015 G0013) (NIL) ((:24)) ((:2 :3 :1) G0046 G0037 G0047 G0038 G0049
 G0040 G0039 G0037))
LOCAL
((NIL) (NIL) ((:20 :19 :21) G0017 G0020 G0019 G0016 G0022 G0020 G0018 G0017) ((:18 :22) G0021 G0023 G0021) (NIL) (NIL) (
NIL) ((:13 :14 :12) G0030 G0042 G0032 G0043 G0031 G0030) ((:10 :11 :9) G0025 G0041 G0028 G0029 G0027 G0025) ((:6 :7 :8)
G0046 G0035 G0036 G0033 G0044 G0035 G0034) ((:16)) ((:23 :17) G0024 G0026 G0024) ((:15 :5 :4) G0015 G0014 G0048 G0045 G0
015 G0013) ((:24)) ((:2 :3 :1) G0046 G0037 G0047 G0038 G0049 G0040 G0039 G0037))
LOCAL
(((:2 :3 :1) G0046 G0037 G0047 G0038 G0049 G0040 G0039 G0037) ((:15 :5 :4) G0015 G0014 G0048 G0045 G0015 G0013) ((:23 :1
7) G0024 G0026 G0024) ((:6 :7 :8) G0046 G0035 G0036 G0033 G0044 G0035 G0034) ((:10 :11 :9) G0025 G0041 G0028 G0029 G0027
 G0025) ((:13 :14 :12) G0030 G0042 G0032 G0043 G0031 G0030) ((:18 :22) G0021 G0023 G0021) ((:20 :19 :21) G0017 G0020 G00
19 G0016 G0022 G0020 G0018 G0017))
LOCAL
SMB
RESULTS
(BODY 1. IS :2 :3 :1)
(BODY 2. IS :15 :5 :4)
(BODY 3. IS :23 :17)
(BODY 4. IS :6 :7 :8)
(BODY 5. IS :10 :11 :9)
(BODY 6. IS :13 :14 :12)
(BODY 7. IS :18 :22)
(BODY 8. IS :20 :19 :21)
NIL
```

RESULTS FOR TOWER

<u>Scene REWOT</u> This scene (see figure 'REWOT') is the same as the scene TOWER (see figure 'TOWER'), but upside down. The program obtains identical results for both scenes (see 'Results for Tower' and 'Results for Rewot'), because SEE does not use information about a body supporting or leaning on another body. For instance, it was not assumed that body :1-2-3 is partially supporting (in figure 'TOWER') body :4-5-15; clearly this assumption fails in case of figure 'REWOT'. But since the assumption is not followed, the program succeeds in both cases. (gives same results).

See table 'ASSUMPTIONS' for suppositions that the program makes or presumptions that it does not need.

The regions :16 and :24 had to be marked as part of the background, following standard practice (cf. 'Input Format').

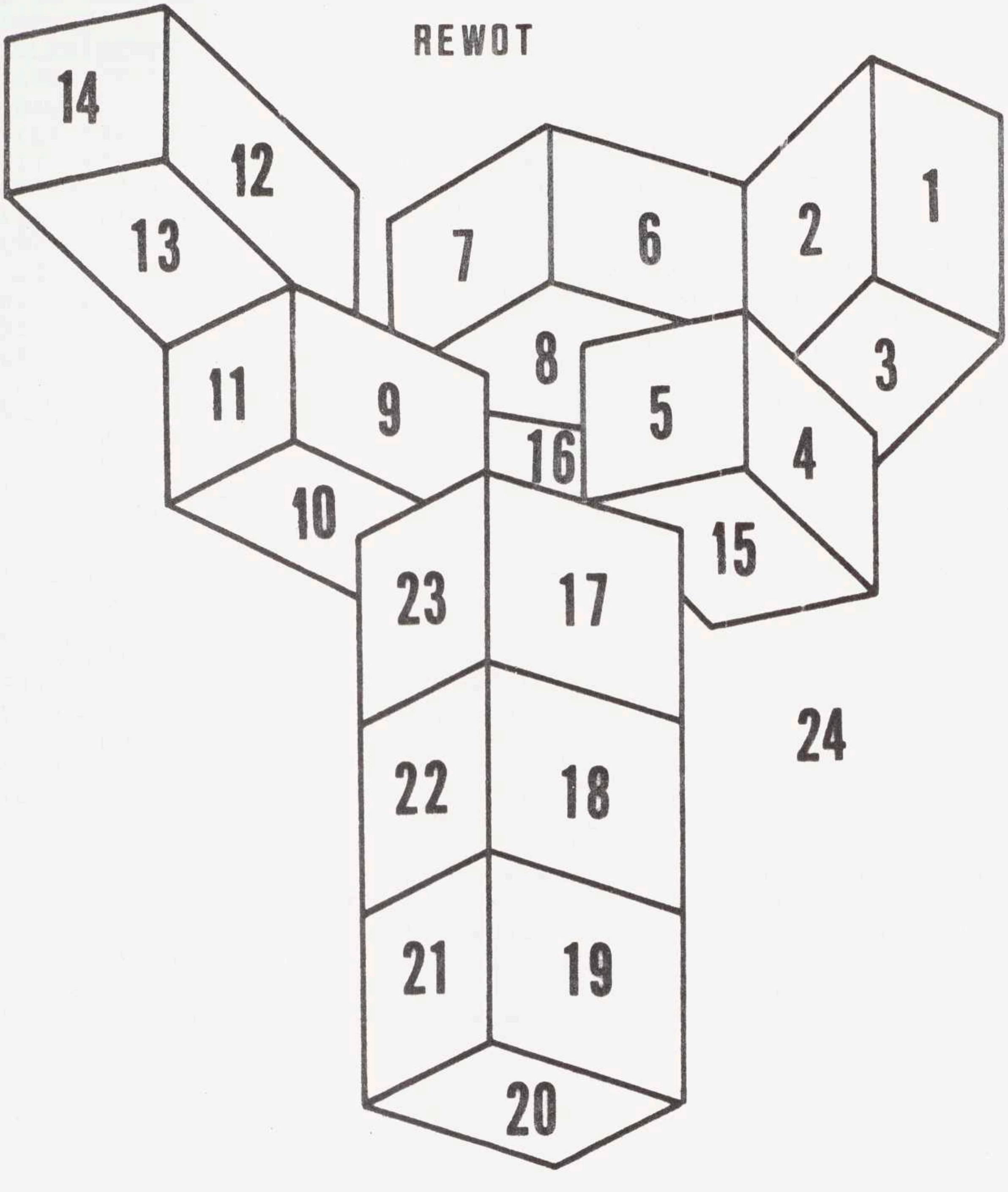


FIGURE 'R E W O T'

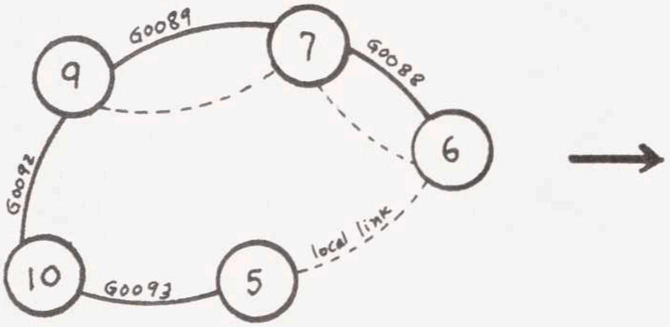
This scene is the same as the scene TOWER, but with Y replaced by 100. - Y, and X replaced by 100. - X : it is upside down. SEE still finds eight bodies.

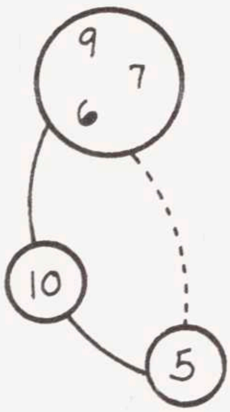
```
SEE 58 ANALYZES REWOT
EVIDENCE
LOCALEVIDENCE
TRIANG
GLOBAL
((NIL) ((:20) G0134 G0133 G0132 G0131) ((:19) G0137 G0135 G0134 G
0131) ((:21) G0137 G0135 G0133 G0132) ((:18) G0138 G0136) ((:22)
G0138 G0136) ((:23) G0141 G0139) ((:10) G0156 G0144 G0143 G0140)
((:11) G0156 G0143 G0142) ((:13) G0157 G0147 G0145) ((:14) G0158
G0157 G0147 G0146) ((:6) G0161 G0159 G0151 G0150 G0148) ((:2) G01
62 G0161 G0155 G0153 G0152) ((:12) G0158 G0146 G0145) ((:7) G0151
 G0149 G0148) ((:9) G0144 G0142 G0140) ((:8) G0159 G0150 G0149) (
(:5) G0160 G0129 G0128) ((:16)) ((:17) G0141 G0139) ((:15) G0163
G0130 G0129) ((:4) G0163 G0160 G0130 G0128) ((:3) G0164 G0162 G01
54 G0153) ((:24)) ((:1) G0164 G0155 G0154 G0152))
((NIL) (NIL) (NIL) ((:20 :19 :21) G0132 G0135 G0134 G0131 G0137 G
0135 G0133 G0132) (NIL) ((:18 :22) G0136 G0138 G0136) (NIL) (NIL)
 (NIL) (NIL) (NIL) (NIL) (NIL) ((:13 :14 :12) G0145 G0157 G0147 G
0158 G0146 G0145) (NIL) ((:10 :11 :9) G0140 G0156 G0143 G0144 G01
42 G0140) ((:6 :7 :8) G0161 G0150 G0151 G0148 G0159 G0150 G0149)
(NIL) ((:16)) ((:23 :17) G0139 G0141 G0139) (NIL) ((:15 :5 :4) G0
130 G0129 G0163 G0160 G0130 G0128) (NIL) ((:24)) ((:2 :3 :1) G016
1 G0152 G0162 G0153 G0164 G0155 G0154 G0152))
LOCAL
((NIL) (NIL) ((:20 :19 :21) G0132 G0135 G0134 G0131 G0137 G0135 G
0133 G0132) ((:18 :22) G0136 G0138 G0136) (NIL) (NIL) (NIL) ((:13
 :14 :12) G0145 G0157 G0147 G0158 G0146 G0145) ((:10 :11 :9) G014
0 G0156 G0143 G0144 G0142 G0140) ((:6 :7 :8) G0161 G0150 G0151 G0
148 G0159 G0150 G0149) ((:16)) ((:23 :17) G0139 G0141 G0139) ((:1
5 :5 :4) G0130 G0129 G0163 G0160 G0130 G0128) ((:24)) ((:2 :3 :1)
 G0161 G0152 G0162 G0153 G0164 G0155 G0154 G0152))
LOCAL
(((:2 :3 :1) G0161 G0152 G0162 G0153 G0164 G0155 G0154 G0152) ((:
15 :5 :4) G0130 G0129 G0163 G0160 G0130 G0128) ((:23 :17) G0139 G
0141 G0139) ((:6 :7 :8) G0161 G0150 G0151 G0148 G0159 G0150 G0149
) ((:10 :11 :9) G0140 G0156 G0143 G0144 G0142 G0140) ((:13 :14 :1
2) G0145 G0157 G0147 G0158 G0146 G0145) ((:18 :22) G0136 G0138 G0
136) ((:20 :19 :21) G0132 G0135 G0134 G0131 G0137 G0135 G0133 G01
32))
LOCAL
SMB
RESULTS
(BODY 1. IS :2 :3 :1)
(BODY 2. IS :15 :5 :4)
(BODY 3. IS :23 :17)
(BODY 4. IS :6 :7 :8)
(BODY 5. IS :10 :11 :9)
(BODY 6. IS :13 :14 :12)
(BODY 7. IS :18 :22)
(BODY 8. IS :20 :19 :21)
NIL
```

RESULTS FOR REWOT

Scene WRIST* The concave objects are properly identified. W places a link between :23 and :4, and another between :30 and :4. CC does not inhibit the link between :17 and :19 ordered by the Arrow NA, because NOSABO was never called, since the first rule of 'ARROW' (page 84) was applied.

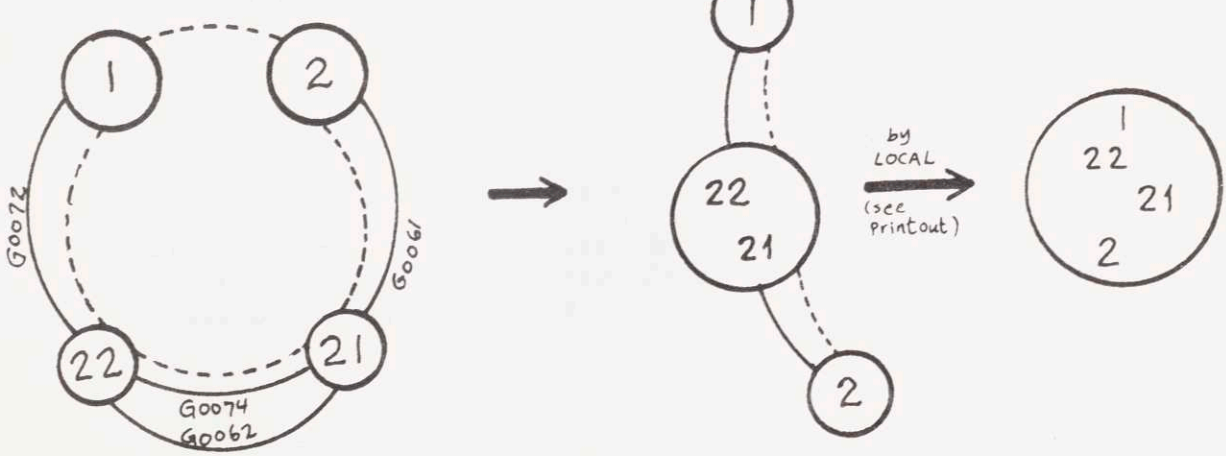
The only mistake was that objects :9-7-6 and :10-5 should be fused and reported as only one. There is a link between :9 and :10 put by heuristic (g) of table 'GLOBAL EVIDENCE'. It is not enough. There is also a weak link between 'Triangles' :5 and :6. OB is not a 'Leg', so there is no weak link between :10 and :5. The situation is as follows (see chains of links in 'RESULTS FOR WRIST*; how to read these chains is explained in page 110 , 'Explanation of the print-out produced by the program'):





:10 and :5 will get joined later by SINGLEBODY.

Almost the same thing occurs with :1-2-22-21, but in this case vertex A produces one strong link between 22 and 21, and vertex R, by heuristic (g) of table 'Global Evidence', also links 22 with 21. This is enough.



WRIST *

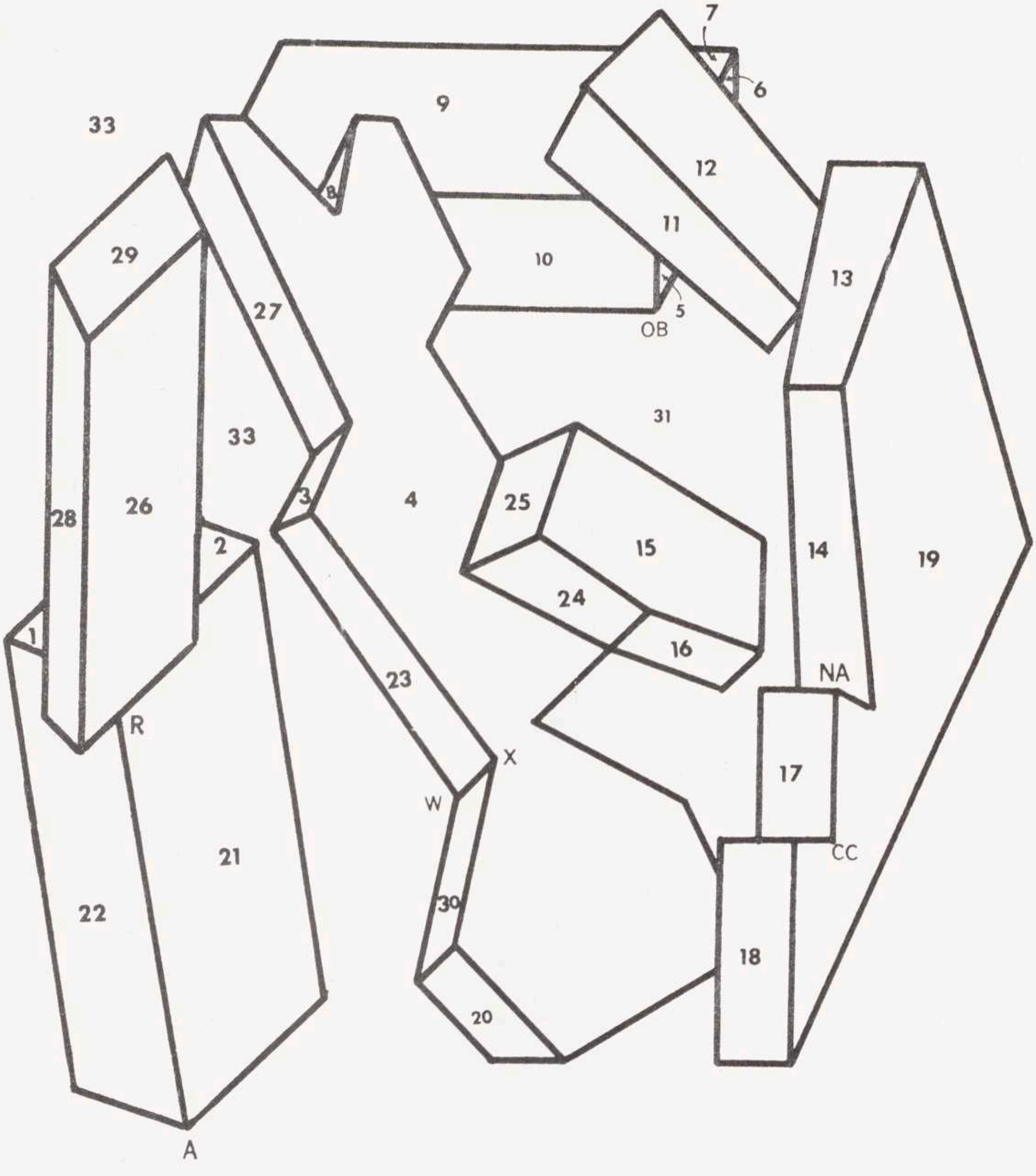


FIGURE 'W R I S T *'

Instead of one, two bodies were found in :9-7-6 and :10-5 Insufficiency of links was the offending reason. All other objects were correctly found.

```
SEE 58 ANALYZES WRIST*
EVIDENCE
(O OF STUDY)
LOCALEVIDENCE
TRIANG
GLOBAL
((NIL) ((:3) G0067 G0065 G0064 G0063) ((:27) G0066 G0065) ((:29) G0071 G0070 G0068) ((:1) G0072) ((:28) G0073 G0071 G006
9 G0068) ((:21) G0074 G0062 G0061) ((:2) G0061) ((:26) G0073 G0070 G0069) ((:22) G0074 G0072 G0062) ((:23) G0077 G0075 G
0064 G0063) ((:30) G0079 G0078 G0077 G0076) ((:20) G0081 G0080 G0079 G0078) ((:18) G0082) ((:14) G0086 G0084) ((:13) G00
87 G0086 G0085 G0084) ((:8) G0091) ((:10) G0093 G0092) ((:5) G0093) ((:11) G0090) ((:25) G0098 G0096 G0095 G0094) ((:24)
 G0101 G0099 G0097 G0096 G0095) ((:16) G0102 G0100 G0099) ((:15) G0102 G0101 G0100 G0098 G0097 G0094) ((:31)) ((:9) G009
2 G0089) ((:4) G0091 G0081 G0080 G0076 G0075 G0067 G0066) ((:17) G0083) ((:19) G0087 G0085 G0083 G0082) ((:6) G0088) ((:
7) G0089 G0088) ((:12) G0090) ((:33)))
((NIL) (NIL) ((:27) G0066 G0065) (NIL) ((:1) G0072) (NIL) (NIL) ((:2) G0061) ((:29 :28 :26) G0073 G0071 G0068 G0073 G007
0 G0069) ((:21 :22) G0062 G0061 G0074 G0072 G0062) (NIL) (NIL) (NIL) ((:18) G0082) (NIL) (NIL) ((:8) G0091) ((:10) G0093
 G0092) ((:5) G0093) ((:11) G0090) (NIL) (NIL) (NIL) ((:16 :25 :24 :15) G0100 G0095 G0099 G0096 G0095 G0102 G0101 G0100
G0098 G0097 G0094) ((:31)) ((:9) G0092 G0089) ((:30 :20 :3 :23 :4) G0077 G0081 G0079 G0078 G0065 G0077 G0064 G0063 G0091
 G0081 G0080 G0076 G0075 G0067 G0066) ((:17) G0083) ((:14 :13 :19) G0087 G0086 G0084 G0087 G0085 G0083 G0082) ((:6) G008
8) ((:7) G0089 G0088) ((:12) G0090) ((:33)));
LOCAL
(LOCAL ASSUMES (:11) (:12) SAME BODY)
(LOCAL ASSUMES (:9) (:7) SAME BODY)
(LOCAL ASSUMES (:6) (:7 :9) SAME BODY)
(LOCAL ASSUMES (:17) (:14 :13 :19) SAME BODY)
(LOCAL ASSUMES (:18) (:14 :13 :19 :17) SAME BODY)
(LOCAL ASSUMES (:1) (:21 :22) SAME BODY)
(LOCAL ASSUMES (:21 :22 :1) (:2) SAME BODY)
LOCAL
((NIL) ((:27) G0066 G0065) ((:2 :21 :22 :1) G0061 G0074 G0062 G0072) (NIL) (NIL) ((:29 :28 :26) G0073 G0071 G0068 G0073
G0070 G0069) (NIL) (NIL) ((:14 :13 :19 :17 :18) G0086 G0084 G0087 G0085 G0083 G0082) (NIL) ((:8) G0091) ((:10) G0093 G00
92) ((:5) G0093) ((:12 :11) G0090) (NIL) ((:16 :25 :24 :15) G0100 G0095 G0099 G0096 G0095 G0102 G0101 G0100 G0098 G0097
G0094) ((:31)) (NIL) ((:30 :20 :3 :23 :4) G0077 G0081 G0079 G0078 G0065 G0077 G0064 G0063 G0091 G0081 G0080 G0076 G0075
G0067 G0066) (NIL) (NIL) ((:7 :9 :6) G0092 G0089 G0088) (NIL) (NIL) ((:33)))
((NIL) (NIL) ((:2 :21 :22 :1) G0061 G0074 G0062 G0072) (NIL) ((:29 :28 :26) G0073 G0071 G0068 G0073 G0070 G0069) (NIL) (
(:14 :13 :19 :17 :18) G0086 G0084 G0087 G0085 G0083 G0082) ((:8) G0091) ((:10) G0093 G0092) ((:5) G0093) ((:12 :11) G009
0) ((:16 :25 :24 :15) G0100 G0095 G0099 G0096 G0095 G0102 G0101 G0100 G0098 G0097 G0094) ((:31)) ((:27 :30 :20 :3 :23 :4
) G0065 G0081 G0079 G0078 G0065 G0077 G0064 G0063 G0091 G0081 G0080 G0076 G0075 G0067 G0066) (NIL) ((:7 :9 :6) G0092 G00
89 G0088) (NIL) ((:33)))
LOCAL
(SINGLEBODY ASSUMES (:10) (:5) SAME BODY)
(SINGLEBODY ASSUMES (:27 :30 :20 :3 :23 :4) (:8) SAME BODY)
(((:7 :9 :6) G0092 G0089 G0088) ((:27 :30 :20 :3 :23 :4 :8) G0065 G0081 G0079 G0078 G0065 G0077 G0064 G0063 G0091 G0081
G0080 G0076 G0075 G0067 G0066) ((:16 :25 :24 :15) G0100 G0095 G0099 G0096 G0095 G0102 G0101 G0100 G0098 G0097 G0094) ((:
12 :11) G0090) (NIL) ((:10 :5) G0093 G0092) (NIL) ((:14 :13 :19 :17 :18) G0086 G0084 G0087 G0085 G0083 G0082) ((:29 :28
:26) G0073 G0071 G0068 G0073 G0070 G0069) ((:2 :21 :22 :1) G0061 G0074 G0062 G0072))
LOCAL
SMB
RESULTS
(BODY 1. IS :7 :9 :6)
(BODY 2. IS :27 :30 :20 :3 :23 :4 :8)
(BODY 3. IS :16 :25 :24 :15)
(BODY 4. IS :12 :11)
(BODY 5. IS :10 :5)
(BODY 6. IS :14 :13 :19 :17 :18)
(BODY 7. IS :29 :28 :26)
(BODY 8. IS :2 :21 :22 :1)
NIL
```

RESULTS FOR WRIST*

Scenes L2 and R2 Two objects are found, as expected.

This scene and L2 form an stereographic pair: two pictures taken from the same scene from slightly different locations, mantaining parallel the optical axes of the cameras, and the same magnification. A program, not yet completed, is designed with the following ideas: Left and right pictures are independently processed by SEE; L2 and R2 in this example. The answers are

ANALYSIS OF L2	ANALYSIS OF R2
(BODY 1. IS :2 :4)	(BODY 1. IS %:1 %:2 %:4)
(BODY 2. IS :1 :5 :3)	(BODY 2. IS %:3 %:6 %:5)

The question is now: Is body :2-:4 the same body as %:1-%:2-%:4, or is it %:3-%:6-%:5 ? It is required, after decomposition of the scene into bodies, to match the left bodies with the right bodies. If this is accomplished, one could then locate the figure in three dimensional space, from the two-dimensional coordinates of the left and right scenes.

In this way it will be known where these objects are located in the "real world".

This "matching" mentioned above is complicated as follows:

== It is possible that the number of objects observed in one view be different from the number in the other.

== On a given object, it is possible that SEE will make a mistake in the left view, but not in the right view; as a consequence, two bodies on the left have to be matched with one on the right.

If the two axes of the camera are on an horizontal plane, a vertex in the left scene and its corresponding vertex in the right scene (if visible) will have the same y-coordinate, such as H in L2 and %I in R2. Other known relations exist, derived from the relative position of the axes of the camera, magnification, etc. See section 'Stereo Perception'.

R 2

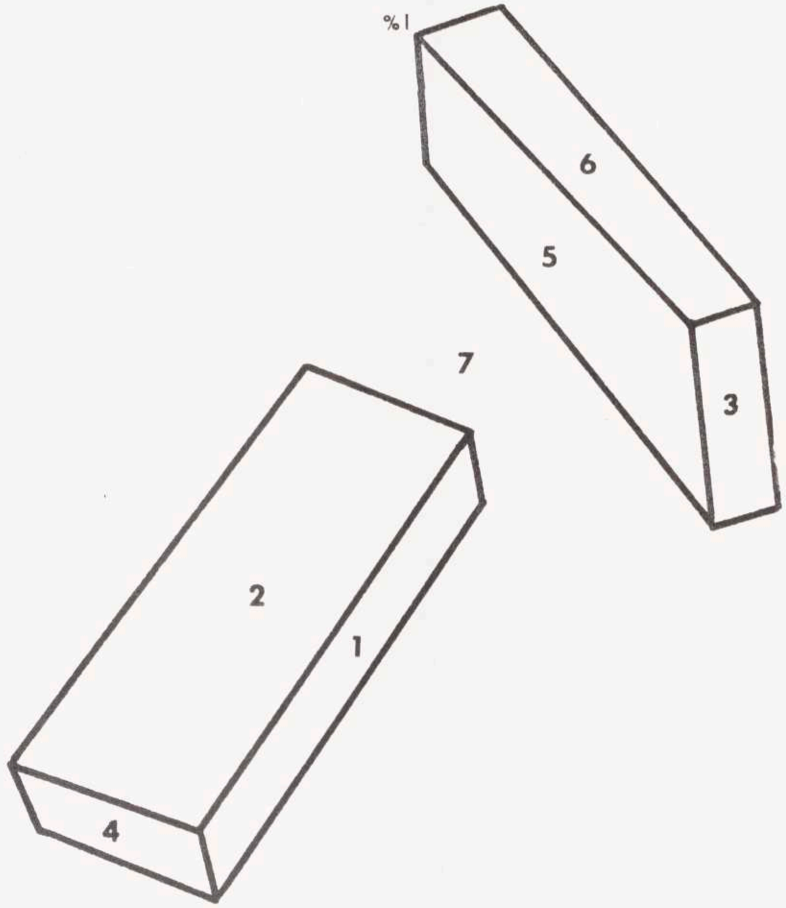


FIGURE "R 2"
Two bricks are found.

```
SEE 58 ANALYZES R2
EVIDENCE
LOCALEVIDENCE
TRIANG
GLOBAL
((NIL) ((X:3) G0009 G0008 G0005 G0004) ((X:6) G0010 G0009 G0007 G0005) ((X:5) G0010 G0008 G0007 G0004) ((X:1) G0014 G001
3 G0012 G0011) ((X:2) G0016 G0015 G0014 G0011) ((X:4) G0016 G0015 G0013 G0012) ((X:7)))
((NIL) (NIL) (NIL) ((X:3 X:6 X:5) G0004 G0009 G0005 G0010 G0008 G0007 G0004) (NIL) (NIL) ((X:1 X:2 X:4) G0012 G0015 G001
4 G0011 G0016 G0015 G0013 G0012) ((X:7)))
LOCAL
((NIL) (NIL) ((X:3 X:6 X:5) G0004 G0009 G0005 G0010 G0008 G0007 G0004) (NIL) ((X:1 X:2 X:4) G0012 G0015 G0014 G0011 G001
6 G0015 G0013 G0012) ((X:7)))
LOCAL
(((X:1 X:2 X:4) G0012 G0015 G0014 G0011 G0016 G0015 G0013 G0012) ((X:3 X:6 X:5) G0004 G0009 G0005 G0010 G0008 G0007 G000
4))
LOCAL
SMB
RESULTS
(BODY 1. IS X:1 X:2 X:4)
(BODY 2. IS X:3 X:6 X:5)
NIL
```

RESULTS FOR R2

L 2

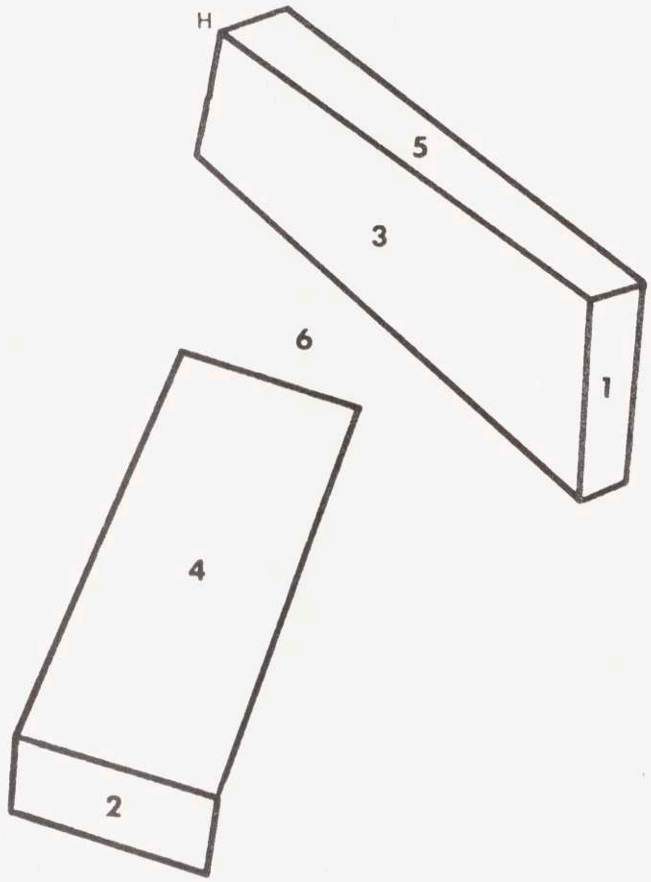


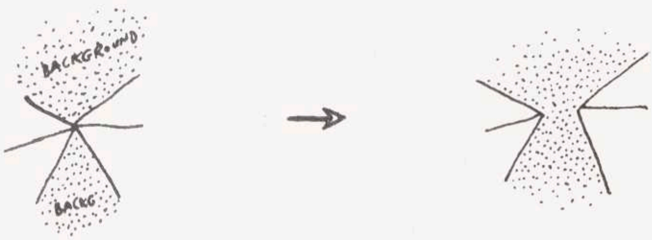
FIGURE 'L 2'

Even if (possibly) a face of object :4-2 is missing, in this case SEE makes the correct identification. Section 'On Noisy Input' deals with imperfect information.

```
SEE 58 ANALYZES L2
EVIDENCE
LOCALEVIDENCE
TRIANG
GLOBAL
((NIL) ((:1) G0009 G0008 G0007 G0004) ((:5) G0010 G0009 G0008 G0006) ((:3) G0010 G0007 G0006 G0004) ((:4) G0011) ((:2) G
0011) ((:6)))
((NIL) (NIL) (NIL) ((:1 :5 :3) G0007 G0010 G0009 G0008 G0010 G0007 G0006 G0004) ((:4) G0011) ((:2) G0011) ((:6)))
LOCAL
(LOCAL ASSUMES (:4) (:2) SAME BODY)
((NIL) (NIL) ((:1 :5 :3) G0007 G0010 G0009 G0008 G0010 G0007 G0006 G0004) ((:2 :4) G0011) (NIL) ((:6)))
LOCAL
(((:2 :4) G0011) ((:1 :5 :3) G0007 G0010 G0009 G0008 G0010 G0007 G0006 G0004))
LOCAL
SMB
RESULTS
(BODY 1. IS :2 :4)
(BODY 2. IS :1 :5 :3)
NIL
```

RESULTS FOR L2

Scene L19 The small triangle :15 just could not get joined with the remainder of the body :16-20-19, and two objects were found. There is a weak link between :15 and :19, but it did not help since there is no link between :15 and :16. What happens is that regions :1, :15, :13 and :22 all meet forming a vertex of type MULTI; this vertex should (in some future version of SEE) be split into two, since both :1 and :37 are the background- The rule for this splitting seems to be



:11 was joined with :4, but isolated from :12-27-5. There are no T-joints between these two nuclei that could give 'hints' (i. e., links) for their unification.

The two large concave objects were properly isolated.

Compare with R19 and WRIST*.

See 'Merged vertices', page 221 in section 'On noisy input.'

L 19

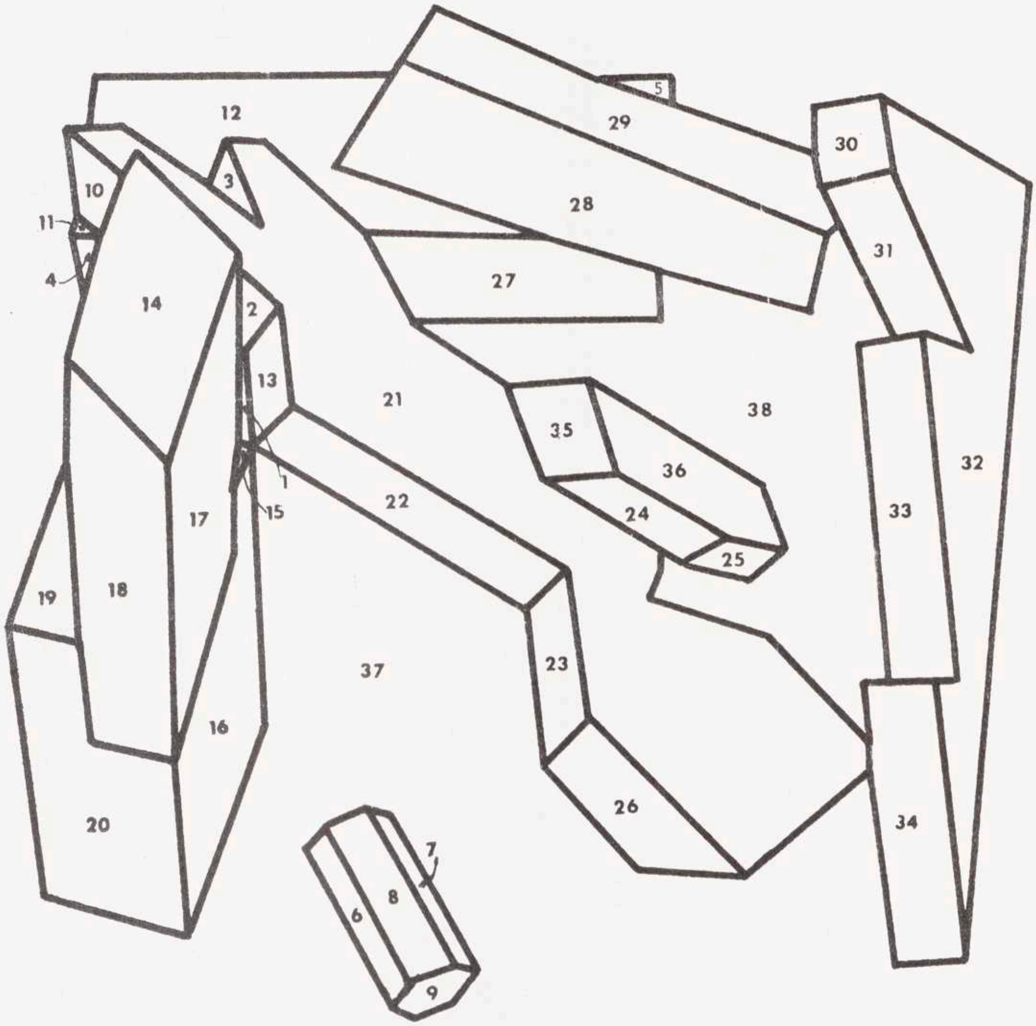


FIGURE 'L 1 9'

It was easy to find :6-7-8-9, the hexagonal prism. :15 was reported as a single object: a mistake. The two big concave objects were appropiately identified.

```
SEE 58 ANALYZES L19
EVIDENCE
LOCALEVIDENCE
TRIANG
GLOBAL
((NIL) ((*32) G0020 G0018 G0017 G0016) ((*33) G0020) ((*34) G0016) ((*9) G0031 G0030 G0028 G0027 G0025 G0024) ((*6) G003
2 G0031 G0030 G0029) ((*7) G0033 G0026 G0025 G0024) ((*8) G0033 G0032 G0029 G0028 G0027 G0026) ((*26) G0034 G0023 G0022
G0021) ((*25) G0039 G0038 G0036 G0035) ((*36) G0043 G0042 G0040 G0039 G0037 G0035) ((*35) G0044 G0043 G0041 G0040) ((*24
) G0044 G0042 G0041 G0038 G0037 G0036) ((*23) G0047 G0045 G0034 G0021) ((*20) G0051 G0049 G0048) ((*19) G0051) ((*18) G0
055 G0054 G0052 G0050) ((*16) G0049 G0048) ((*15)) ((*22) G0056 G0046 G0045) ((*38)) ((*31) G0059 G0019) ((*30) G0059 G0
019 G0018 G0017) ((*5) G0061 G0057) ((*29) G0060) ((*27) G0062 G0058 G0057) ((*28) G0060) ((*3) G0063) ((*13) G0066 G006
4 G0056) ((*1)) ((*2) G0069 G0067 G0065 G0064) ((*17) G0068 G0055 G0053 G0050) ((*4) G0070) ((*14) G0068 G0054 G0053 G00
52) ((*11) G0070) ((*10) G0069) ((*21) G0067 G0066 G0065 G0063 G0047 G0046 G0023 G0022) ((*12) G0062 G0061 G0058) ((*37)
))
((NIL) (NIL) ((*33) G0020) ((*34) G0016) (NIL) (NIL) (NIL) ((*9 *6 *7 *8) G0027 G0032 G0031 G0030 G0033 G0025 G0024 G003
3 G0032 G0029 G0028 G0027 G0026) (NIL) (NIL) (NIL) (NIL) ((*25 *36 *35 *24) G0036 G0039 G0035 G0043 G0040 G0044 G0042 G0
041 G0038 G0037 G0036) (NIL) (NIL) ((*19) G0051) (NIL) ((*20 *16) G0051 G0048 G0049 G0048) ((*15)) (NIL) ((*38)) (NIL) (
(*31 *32 *30) G0019 G0020 G0016 G0059 G0019 G0018 G0017) ((*5) G0061 G0057) ((*29) G0060) (NIL) ((*28) G0060) ((*3) G006
3) (NIL) ((*1)) (NIL) (NIL) ((*4) G0070) ((*18 *17 *14) G0052 G0055 G0050 G0068 G0054 G0053 G0052) ((*11) G0070) ((*10)
G0069) ((*2 *13 *22 *26 *23 *21) G0069 G0065 G0064 G0056 G0045 G0034 G0021 G0067 G0066 G0065 G0063 G0047 G0046 G0023 G00
22) ((*27 *12) G0058 G0057 G0062 G0061 G0058) ((*37)))
((NIL) ((*33) G0020) ((*34) G0016) (NIL) ((*9 *6 *7 *8) G0027 G0032 G0031 G0030 G0033 G0025 G0024 G0033 G0032 G0029 G002
8 G0027 G0026) (NIL) (NIL) ((*25 *36 *35 *24) G0036 G0039 G0035 G0043 G0040 G0044 G0042 G0041 G0038 G0037 G0036) (NIL) (
(*19) G0051) ((*20 *16) G0051 G0048 G0049 G0048) ((*15)) ((*38)) ((*31 *32 *30) G0019 G0020 G0016 G0059 G0019 G0018 G001
7) (NIL) ((*29) G0060) ((*28) G0060) ((*3) G0063) ((*1)) (NIL) ((*4) G0070) ((*18 *17 *14) G0052 G0055 G0050 G0068 G0054
 G0053 G0052) ((*11) G0070) ((*10) G0069) ((*2 *13 *22 *26 *23 *21) G0069 G0065 G0064 G0056 G0045 G0034 G0021 G0067 G006
6 G0065 G0063 G0047 G0046 G0023 G0022) ((*5 *27 *12) G0057 G0057 G0062 G0061 G0058) ((*37)))
LOCAL
(LOCAL ASSUMES (*10) (*2 *13 *22 *26 *23 *21) SAME BODY)
(LOCAL ASSUMES (*20 *16) (*19) SAME BODY)
(LOCAL ASSUMES (*31 *32 *30) (*33) SAME BODY)
(LOCAL ASSUMES (*33 *31 *32 *30) (*34) SAME BODY)
((NIL) (NIL) (NIL) ((*9 *6 *7 *8) G0027 G0032 G0031 G0030 G0033 G0025 G0024 G0033 G0032 G0029 G0028 G0027 G0026) (NIL) (
(*25 *36 *35 *24) G0036 G0039 G0035 G0043 G0040 G0044 G0042 G0041 G0038 G0037 G0036) (NIL) ((*19 *20 *16) G0051 G0049 G0
048) ((*15)) ((*38)) ((*34 *33 *31 *32 *30) G0019 G0020 G0016 G0059 G0019 G0018 G0017) ((*29) G0060) ((*28) G0060) ((*3)
 G0063) ((*1)) ((*4) G0070) ((*18 *17 *14) G0052 G0055 G0050 G0068 G0054 G0053 G0052) ((*11) G0070) ((*2 *13 *22 *26 *23
 *21 *10) G0065 G0064 G0056 G0045 G0034 G0021 G0067 G0066 G0065 G0063 G0047 G0046 G0023 G0022 G0069) (NIL) ((*5 *27 *12)
 G0057 G0057 G0062 G0061 G0058) ((*37)))
LOCAL
(SINGLEBODY ASSUMES (*4) (*11) SAME BODY)
(SINGLEBODY ASSUMES (*2 *13 *22 *26 *23 *21 *10) (*3) SAME BODY)
(SINGLEBODY ASSUMES (*29) (*28) SAME BODY)
(((*5 *27 *12) G0057 G0057 G0062 G0061 G0058) ((*2 *13 *22 *26 *23 *21 *10 *3) G0065 G0064 G0056 G0045 G0034 G0021 G0067
 G0066 G0065 G0063 G0047 G0046 G0023 G0022 G0069) (NIL) ((*18 *17 *14) G0052 G0055 G0050 G0068 G0054 G0053 G0052) ((*4 *
11) G0070) (NIL) (NIL) ((*29 *28) G0060) ((*34 *33 *31 *32 *30) G0019 G0020 G0016 G0059 G0019 G0018 G0017) ((*15)) ((*19
 *20 *16) G0051 G0049 G0048) ((*25 *36 *35 *24) G0036 G0039 G0035 G0043 G0040 G0044 G0042 G0041 G0038 G0037 G0036) ((*9
 *6 *7 *8) G0027 G0032 G0031 G0030 G0033 G0025 G0024 G0033 G0032 G0029 G0028 G0027 G0026))
 LOCAL
 SMB
 RESULTS
 (THE FIRST 1. BODIES ARE ((*15)))
 (BODY 2. IS *5 *27 *12)
 (BODY 3. IS *2 *13 *22 *26 *23 *21 *10 *3)
 (BODY 4. IS *18 *17 *14)
 (BODY 5. IS *4 *11)
 (BODY 6. IS *29 *28)
 (BODY 7. IS *34 *33 *31 *32 *30)
 (BODY 8. IS *19 *20 *16)
 (BODY 9. IS *25 *36 *35 *24)
 (BODY 10. IS *9 *6 *7 *8)
 NIL
```

RESULTS FOR L19

Scene R19 As in L19, here the triangle :27 is detached from :5-32-33, two bodies being reported. There is no strong link between :27 and :33. There is a weak link between :27 and :5, because both are 'triangles' facing each other, but that is not enough. A weak link is never enough.

All other bodies are properly found, including :10-16-2-3.

Vertex RA, of course, contributes with no links. The situation could change if we discover that RA is a false vertex, that is, one composed by the merge of two genuine ones. There is enough enformation, I think, since :34 and :37 are background, and this will suggest a way to "divide" vertex RA into two simpler ones. This idea of dividing vertices of type MULTI into simpler ones should be applied with caution, since there will be genuine vertex of type MULTI (which should not be split). The main use of this technique will be for helping single regions to join some other body, a task performed now, not too satisfactorially, by SMB.

SUGGESTION

Compare with L19 and WRIST*.

See 'merged vertices', page .

R 19

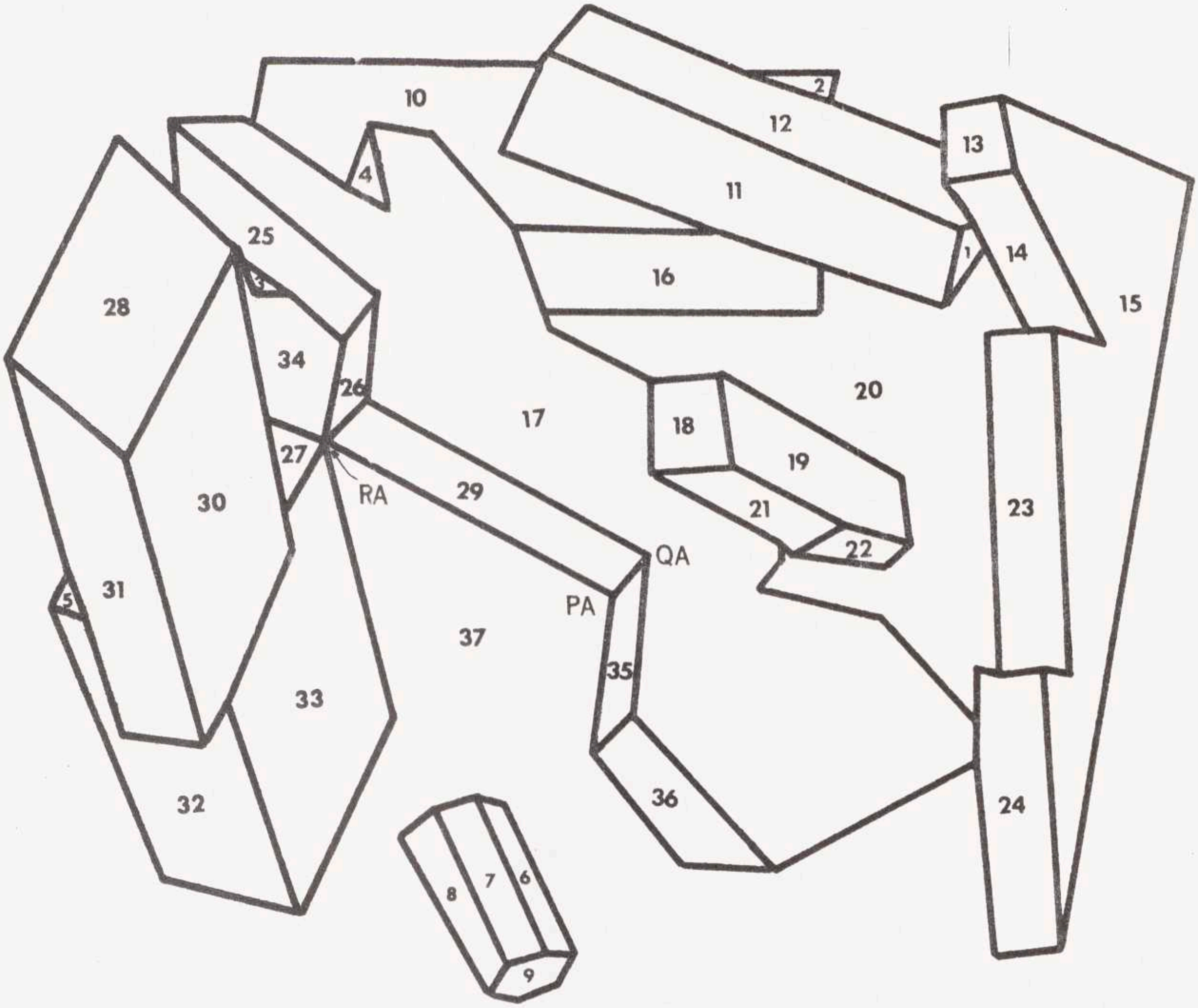


FIGURE 'R 1 9'

:27 was separated from :33-32-5 All other objects were correctly found.

```
SEE 58 ANALYZES R19
EVIDENCE
LOCALEVIDENCE
TRIANG
GLOBAL
((NIL) ((X:22) G0020 G0019 G0018 G0016) ((X:24) G0023 G0022) ((X:23) G0024) ((X:1) G0027 G0026 G0025) ((X:14) G0031 G003
0) ((X:15) G0032 G0029 G0024 G0023 G0022) ((X:13) G0032 G0031 G0030 G0029) ((X:2) G0034 G0033) ((X:12) G0035 G0028 G0027
) ((X:11) G0035 G0028 G0026 G0025) ((X:19) G0040 G0039 G0037 G0019 G0017 G0016) ((X:21) G0041 G0039 G0038 G0020 G0018 G0
017) ((X:18) G0041 G0040 G0038 G0037) ((X:27)) ((X:29) G0044 G0043 G0042) ((X:35) G0048 G0046 G0045 G0043) ((X:36) G0048
 G0047 G0046 G0021) ((X:6) G0051 G0050 G0049 G0015) ((X:9) G0056 G0055 G0053 G0052 G0050 G0049) ((X:7) G0057 G0054 G0053
 G0052 G0051 G0015) ((X:8) G0057 G0056 G0055 G0054) ((X:33) G0059 G0058) ((X:32) G0061 G0059 G0058) ((X:5) G0061) ((X:31
) G0064 G0063 G0062 G0060) ((X:30) G0066 G0065 G0064 G0060) ((X:3) G0067) ((X:34)) ((X:26) G0070 G0069 G0042) ((X:20)) (
(X:16) G0071 G0067 G0036 G0033) ((X:4) G0072) ((X:10) G0071 G0036 G0034) ((X:17) G0072 G0069 G0068 G0047 G0045 G0044 G00
21) ((X:25) G0070 G0068) ((X:28) G0066 G0065 G0063 G0062) ((X:37)))
((NIL) (NIL) (NIL) ((X:23) G0024) (NIL) (NIL) (NIL) ((X:24 X:15 X:14 X:13) G0032 G0024 G0023 G0022 G0032 G0031 G0030 G00
29) ((X:2) G0034 G0033) (NIL) ((X:12 X:1 X:11) G0028 G0027 G0035 G0028 G0026 G0025) (NIL) (NIL) ((X:22 X:19 X:21 X:18) G
0040 G0019 G0016 G0039 G0020 G0018 G0017 G0041 G0040 G0038 G0037) ((X:27)) ((X:29) G0044 G0043 G0042) (NIL) (NIL) (NIL)
(NIL) (NIL) ((X:6 X:9 X:7 X:8) G0056 G0052 G0050 G0049 G0054 G0053 G0052 G0051 G0015 G0057 G0056 G0055 G0054) (NIL) ((X:
33 X:32) G0058 G0061 G0059 G0058) ((X:5) G0061) (NIL) (NIL) ((X:3) G0067) ((X:34)) ((X:26) G0070 G0069 G0042) ((X:20)) (
NIL) ((X:4) G0072) ((X:16 X:10) G0067 G0033 G0071 G0036 G0034) ((X:35 X:36 X:17) G0045 G0043 G0048 G0046 G0072 G0069 G00
68 G0047 G0045 G0044 G0021) ((X:25) G0070 G0068) ((X:31 X:30 X:28) G0062 G0065 G0064 G0060 G0066 G0065 G0063 G0062) ((X:
37)))
((NIL) (NIL) ((X:23) G0024) (NIL) ((X:24 X:15 X:14 X:13) G0032 G0024 G0023 G0022 G0032 G0031 G0030 G0029) (NIL) ((X:12 X
:1 X:11) G0028 G0027 G0035 G0028 G0026 G0025) (NIL) ((X:22 X:19 X:21 X:18) G0040 G0019 G0016 G0039 G0020 G0018 G0017 G00
41 G0040 G0038 G0037) ((X:27)) (NIL) (NIL) (NIL) ((X:6 X:9 X:7 X:8) G0056 G0052 G0050 G0049 G0054 G0053 G0052 G0051 G001
5 G0057 G0056 G0055 G0054) ((X:33 X:32) G0058 G0061 G0059 G0058) ((X:5) G0061) (NIL) ((X:3) G0067) ((X:34)) (NIL) ((X:20
)) ((X:4) G0072) ((X:2 X:16 X:10) G0033 G0067 G0033 G0071 G0036 G0034) (NIL) ((X:26 X:29 X:35 X:36 X:17 X:25) G0042 G004
2 G0043 G0048 G0046 G0072 G0069 G0047 G0045 G0044 G0021 G0070 G0068) ((X:31 X:30 X:28) G0062 G0065 G0064 G0060 G0066 G00
65 G0063 G0062) ((X:37)))
LOCAL
(LOCAL ASSUMES (X:33 X:32) (X:5) SAME BODY)
(LOCAL ASSUMES (X:23) (X:24 X:15 X:14 X:13) SAME BODY)
((NIL) ((X:24 X:15 X:14 X:13 X:23) G0024 G0023 G0022 G0032 G0031 G0030 G0029 G0024) (NIL) ((X:12 X:1 X:11) G0028 G0027 G
0035 G0028 G0026 G0025) ((X:22 X:19 X:21 X:18) G0040 G0019 G0016 G0039 G0020 G0018 G0017 G0041 G0040 G0038 G0037) ((X:27
)) (NIL) ((X:6 X:9 X:7 X:8) G0056 G0052 G0050 G0049 G0054 G0053 G0052 G0051 G0015 G0057 G0056 G0055 G0054) ((X:5 X:33 X:
32) G0058 G0061 G0059 G0058) (NIL) ((X:3) G0067) ((X:34)) ((X:20)) ((X:4) G0072) ((X:2 X:16 X:10) G0033 G0067 G0033 G007
1 G0036 G0034) ((X:26 X:29 X:35 X:36 X:17 X:25) G0042 G0042 G0043 G0048 G0046 G0072 G0069 G0047 G0045 G0044 G0021 G0070
G0068) ((X:31 X:30 X:28) G0062 G0065 G0064 G0060 G0066 G0065 G0063 G0062) ((X:37)))
LOCAL
(SINGLEBODY ASSUMES (X:26 X:29 X:35 X:36 X:17 X:25) (X:4) SAME BODY)
(SINGLEBODY ASSUMES (X:2 X:16 X:10) (X:3) SAME BODY)
(((X:31 X:30 X:28) G0062 G0065 G0064 G0060 G0066 G0065 G0063 G0062) ((X:26 X:29 X:35 X:36 X:17 X:25 X:4) G0042 G0042 G00
43 G0048 G0046 G0072 G0069 G0047 G0045 G0044 G0021 G0070 G0068) ((X:2 X:16 X:10 X:3) G0033 G0067 G0033 G0071 G0036 G0034
) (NIL) (NIL) ((X:5 X:33 X:32) G0058 G0061 G0059 G0058) ((X:6 X:9 X:7 X:8) G0056 G0052 G0050 G0049 G0054 G0053 G0052 G00
51 G0015 G0057 G0056 G0055 G0054) ((X:27)) ((X:22 X:19 X:21 X:18) G0040 G0019 G0016 G0039 G0020 G0018 G0017 G0041 G0040
G0038 G0037) ((X:12 X:1 X:11) G0028 G0027 G0035 G0028 G0026 G0025) ((X:24 X:15 X:14 X:13 X:23) G0024 G0023 G0022 G0032 G
0031 G0030 G0029 G0024))
LOCAL
SMB
RESULTS
(THE FIRST 1. BODIES ARE ((X:27)))
(BODY 2. IS X:31 X:30 X:28)
(BODY 3. IS X:26 X:29 X:35 X:36 X:17 X:25 X:4)
(BODY 4. IS X:2 X:16 X:10 X:3)
(BODY 5. IS X:5 X:33 X:32)
(BODY 6. IS X:6 X:9 X:7 X:8)
(BODY 7. IS X:22 X:19 X:21 X:18)
(BODY 8. IS X:12 X:1 X:11)
(BODY 9. IS X:24 X:15 X:14 X:13 X:23)
NIL
```

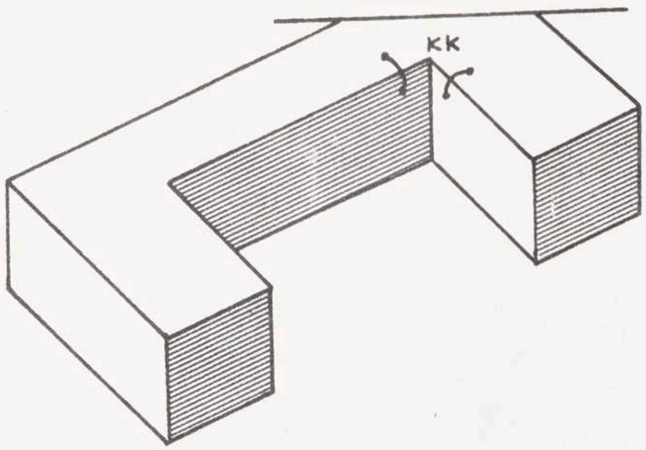
RESULTS FOR R19

Scene CORN The pyramid :8-9-10 was easily identified because a vertex of type PEAK produces many links. In the bottom, bodies :1-2-3-4 and :12-13-11 were separated, because the fork between :4 and :12 has the background as a region, and did not contribute with any links. Certainly, this is a possible interpretation. Another interpretation is to regard the object :1-2-3-4-11-12-13 as a prism with the shape of a "C".

SINGLEBODY was needed to join :4 with :2-3-1, the only link being placed by heuristic (g) of table 'GLOBAL EVIDENCE.'

The program knows that :22 is the background.

If we could see the hidden vertex KK (if it indeed exists), two links would be put and we will have had one body:



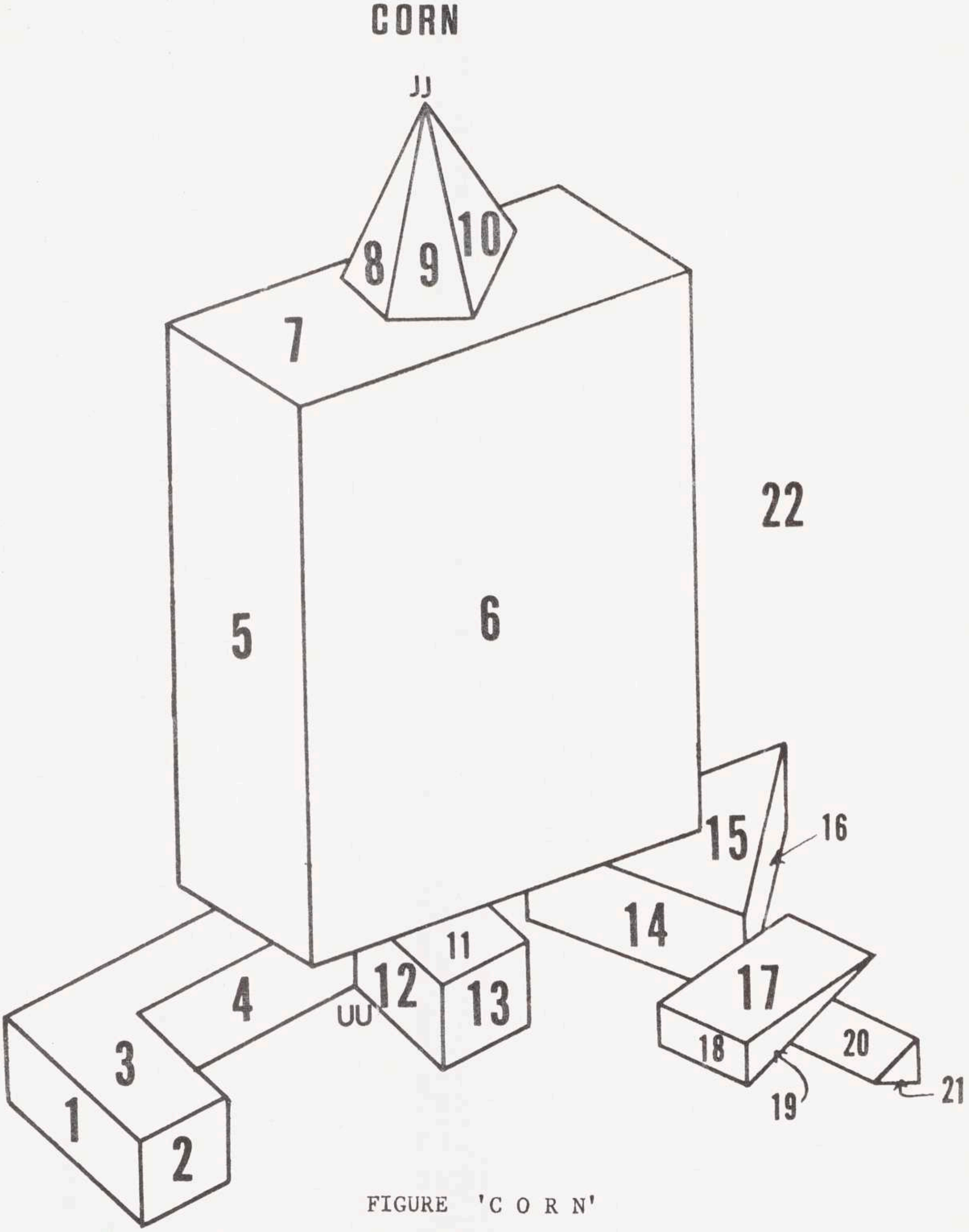


FIGURE 'C O R N'

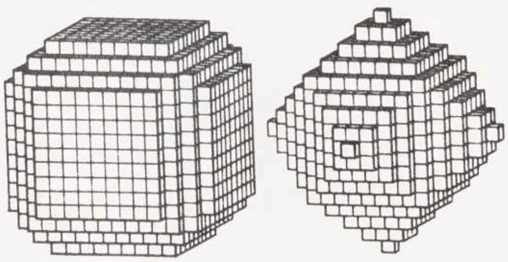
The pyramid at the top was identified properly. Two bodies were found at the bottom, which is a plausible interpretation: :1=2=3=4 and :11=12=13.

```
SEE 58 ANALYZES CORN
EVIDENCE
LOCALEVIDENCE
TRIANG
GLOBAL
((NIL) ((:2) G0019 G0018 G0017 G0015) ((:8) G0026 G0024) ((:9) G0027 G0026 G0025 G0024) ((:10) G0027 G0025) ((:7) G0028
G0023 G0022 G0020) ((:15) G0031 G0030 G0013) ((:16) G0032 G0030 G0029 G0013) ((:21) G0035 G0034) ((:20) G0036 G0035 G003
4) ((:19) G0039 G0038 G0037 G0033) ((:18) G0041 G0040 G0038 G0037) ((:17) G0041 G0040 G0039 G0033) ((:14) G0036 G0032 G0
031 G0029) ((:13) G0044 G0043 G0042 G0014) ((:11) G0046 G0045 G0044 G0042) ((:12) G0046 G0045 G0043 G0014) ((:6) G0047 G
0028 G0022 G0021) ((:4) G0048) ((:5) G0047 G0023 G0021 G0020) ((:22)) ((:3) G0049 G0048 G0018 G0016 G0015) ((:1) G0049 G
0019 G0017 G0016))
((NIL) (NIL) (NIL) (NIL) ((:8 :9 :10) G0027 G0026 G0024 G0027 G0025) (NIL) (NIL) (NIL) (NIL) ((:21 :20) G0034 G0036 G003
5 G0034) (NIL) (NIL) ((:19 :18 :17) G0037 G0041 G0038 G0037 G0041 G0040 G0039 G0033) ((:15 :16 :14) G0013 G0030 G0013 G0
036 G0032 G0031 G0029) (NIL) (NIL) ((:13 :11 :12) G0014 G0045 G0044 G0042 G0046 G0045 G0043 G0014) (NIL) ((:4) G0048) ((
:7 :6 :5) G0020 G0028 G0022 G0047 G0023 G0021 G0020) ((:22)) (NIL) ((:2 :3 :1) G0017 G0048 G0018 G0015 G0049 G0019 G0017
 G0016))
LOCAL
((NIL) (NIL) ((:8 :9 :10) G0027 G0026 G0024 G0027 G0025) (NIL) (NIL) ((:21 :20) G0034 G0036 G0035 G0034) (NIL) ((:19 :18
 :17) G0037 G0041 G0038 G0037 G0041 G0040 G0039 G0033) ((:15 :16 :14) G0013 G0030 G0013 G0036 G0032 G0031 G0029) (NIL) (
(:13 :11 :12) G0014 G0045 G0044 G0042 G0046 G0045 G0043 G0014) ((:4) G0048).((:7 :6 :5) G0020 G0028 G0022 G0047 G0023 G0
021 G0020) ((:22)) ((:2 :3 :1) G0017 G0048 G0018 G0015 G0049 G0019 G0017 G0016))
LOCAL
(SINGLEBODY ASSUMES (:2 :3 :1) (:4) SAME BODY)
(((:2 :3 :1 :4) G0017 G0048 G0018 G0015 G0049 G0019 G0017 G0016) ((:7 :6 :5) G0020 G0028 G0022 G0047 G0023 G0021 G0020)
(NIL) ((:13 :11 :12) G0014 G0045 G0044 G0042 G0046 G0045 G0043 G0014) ((:15 :16 :14) G0013 G0030 G0013 G0036 G0032 G0031
 G0029) ((:19 :18 :17) G0037 G0041 G0038 G0037 G0041 G0040 G0039 G0033) ((:21 :20) G0034 G0036 G0035 G0034) ((:8 :9 :10)
 G0027 G0026 G0024 G0027 G0025))
LOCAL
SMB
RESULTS
(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)
(BODY 5. IS :19 :18 :17)
(BODY 6. IS :21 :20)
(BODY 7. IS :8 :9 :10)
NIL
```

RESULTS FOR CORN

<u>Scene L9</u> Here the tolerances SINTO and COLTO that allow for "sloppy parallelism" have made T's out of NA and PA. Therefore, these vertices do not contribute any links for :1. More over, the "T" PA inhibits the link suggested by QA between :1 and :8. That being all, :1 gets reported as a single body (see next page).

By decreasing the tolerances, correct identification is possible (see the correct identification in page 155).
See 'Tolerances in collinearity and parallelism', page 215.



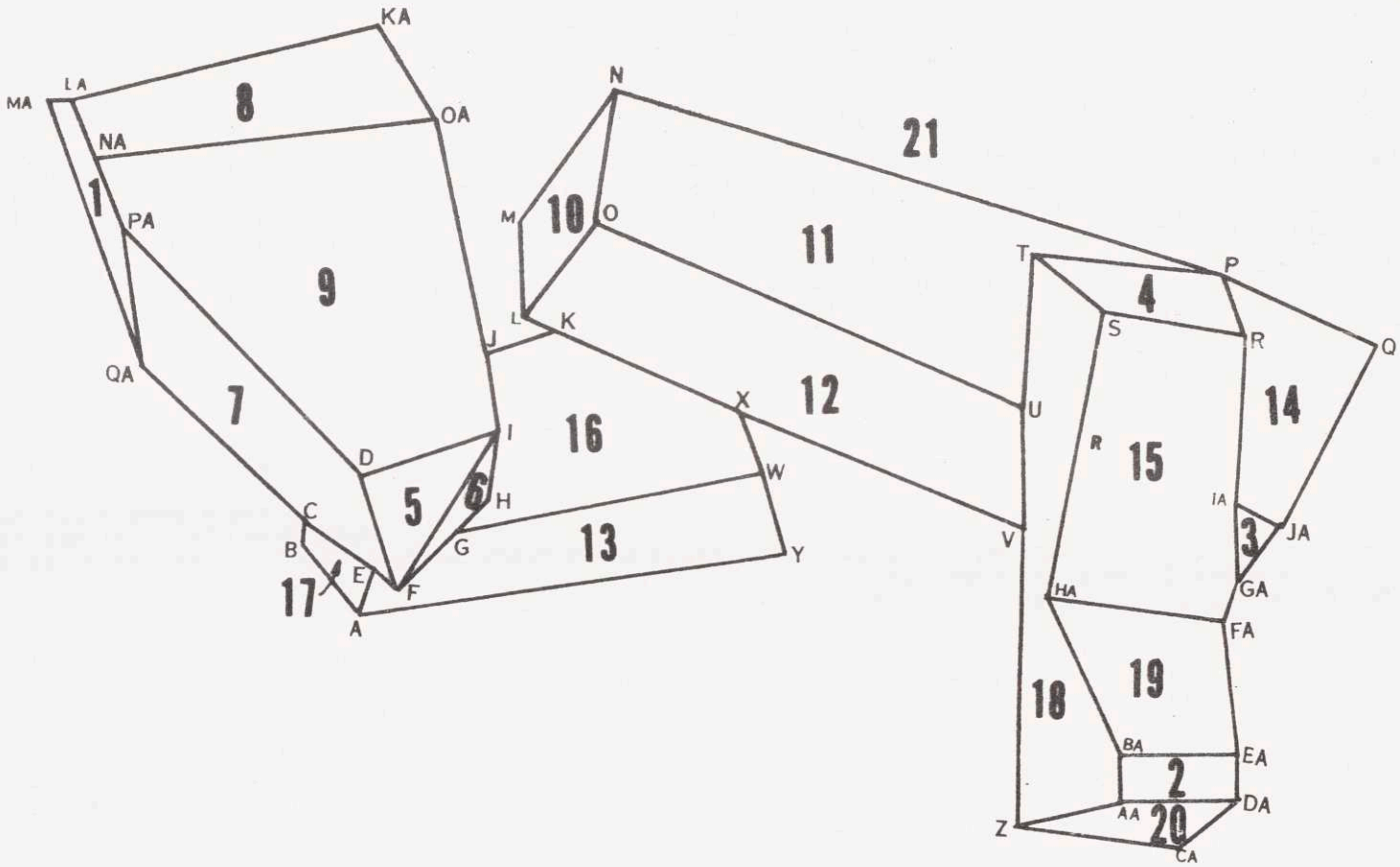


FIGURE 'L 9''

Four bodies are identified. Body :1-8-9-7-5-6 gives some problems.

L 9

See also next page.

```
SEE 58 ANALYZES L9
EVIDENCE
LOCALEVIDENCE
TRIANG
GLOBAL
((NIL) ((*1)) ((*8) G0012) ((*3) G0026 G0013) ((*19) G0019 G0015 G0014) ((*2) G0021 G0020 G0019 G0018 G0017) ((*20) G002
3 G0022 G0020 G0017) ((*18) G0030 G0028 G0023 G0022 G0021 G0018 G0016 G0015) ((*15) G0031 G0029 G0016 G0014) ((*14) G002
7 G0013) ((*4) G0031 G0030 G0029 G0028) ((*11) G0035 G0033 G0032 G0027 G0025) ((*10) G0036 G0035 G0034 G0033) ((*12) G00
36 G0034 G0032 G0026 G0025) ((*16) G0024) ((*6) G0039 G0038) ((*5) G0044 G0043 G0040 G0039 G0038 G0037) ((*9) G0044 G004
2 G0037 G0012) ((*7) G0043 G0042 G0040) ((*17) G0045 G0041) ((*13) G0045 G0041 G0024) ((*21)))
((NIL) ((*1)) ((*8) G0012) ((*3) G0026 G0013) ((*19) G0019 G0015 G0014) (NIL) (NIL) (NIL) (NIL) ((*14) G0027 G0013) ((*1
5 *2 *20 *18 *4) G0029 G0014 G0019 G0020 G0017 G0023 G0022 G0021 G0018 G0016 G0015 G0031 G0030 G0029 G0028) (NIL) (NIL)
((*11 *10 *12) G0032 G0027 G0036 G0035 G0033 G0036 G0034 G0032 G0026 G0025) ((*16) G0024) (NIL) (NIL) (NIL) ((*6 *5 *9 *
7) G0043 G0039 G0038 G0044 G0037 G0012 G0043 G0042 G0040) (NIL) ((*17 *13) G0041 G0045 G0041 G0024) ((*21)))
((NIL) ((*1)) ((*8) G0012) ((*3) G0026 G0013) (NIL) (NIL) (NIL) ((*14) G0027 G0013) ((*19 *15 *2 *20 *18 *4) G0015 G0029
 G0014 G0019 G0020 G0017 G0023 G0022 G0021 G0018 G0016 G0015 G0031 G0030 G0029 G0028) (NIL) ((*11 *10 *12) G0032 G0027 G
0036 G0035 G0033 G0036 G0034 G0032 G0026 G0025) ((*16) G0024) (NIL) ((*6 *5 *9 *7) G0043 G0039 G0038 G0044 G0037 G0012 G
0043 G0042 G0040) ((*17 *13) G0041 G0045 G0041 G0024) ((*21)))
LOCAL
(LOCAL ASSUMES (*3) (*14) SAME BODY)
(LOCAL ASSUMES (*8) (*6 *5 *9 *7) SAME BODY)
((NIL) ((*1)) ((*6 *5 *9 *7 *8) G0039 G0038 G0044 G0037 G0043 G0042 G0040 G0012) ((*14 *3) G0027 G0026 G0013) (NIL) (NIL
) ((*19 *15 *2 *20 *18 *4) G0015 G0029 G0014 G0019 G0020 G0017 G0023 G0022 G0021 G0018 G0016 G0015 G0031 G0030 G0029 G00
28) ((*11 *10 *12) G0032 G0027 G0036 G0035 G0033 G0036 G0034 G0032 G0026 G0025) ((*16) G0024) (NIL) ((*17 *13) G0041 G00
45 G0041 G0024) ((*21)))
((NIL) ((*1)) ((*6 *5 *9 *7 *8) G0039 G0038 G0044 G0037 G0043 G0042 G0040 G0012) (NIL) (NIL) ((*19 *15 *2 *20 *18 *4) G0
015 G0029 G0014 G0019 G0020 G0017 G0023 G0022 G0021 G0018 G0016 G0015 G0031 G0030 G0029 G0028) ((*14 *3 *11 *10 *12) G00
26 G0013 G0027 G0035 G0033 G0036 G0034 G0032 G0026 G0025) ((*16) G0024) ((*17 *13) G0041 G0045 G0041 G0024) ((*21)))
LOCAL
(SINGLEBODY ASSUMES (*17 *13) (*16) SAME BODY)
(((*17 *13 *16) G0041 G0045 G0041 G0024) (NIL) ((*14 *3 *11 *10 *12) G0026 G0013 G0027 G0035 G0033 G0036 G0034 G0032 G00
26 G0025) ((*19 *15 *2 *20 *18 *4) G0015 G0029 G0014 G0019 G0020 G0017 G0023 G0022 G0021 G0018 G0016 G0015 G0031 G0030 G
0029 G0028) ((*6 *5 *9 *7 *8) G0039 G0038 G0044 G0037 G0043 G0042 G0040 G0012) ((*1)))
LOCAL
SMB
RESULTS
(THE FIRST 1. BODIES ARE ((*1)))
(BODY 2. IS *17 *13 *16)
(BODY 3. IS *14 *3 *11 *10 *12)
(BODY 4. IS *19 *15 *2 *20 *18 *4)
(BODY 5. IS *6 *5 *9 *7 *8)
NIL
```

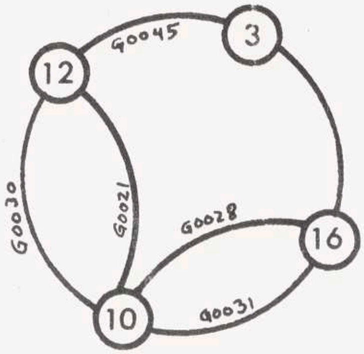
RESULTS FOR L9

Smaller values for SINTO and
COLTO, the parameters for
parallelism and colinearity,
produce correct answers for L9.
Compare with previous page.

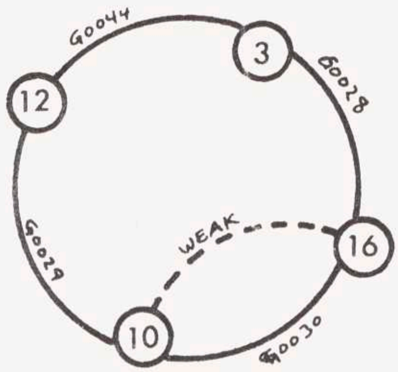
```
SEE 58 ANALYZES L9
EVIDENCE
LOCALEVIDENCE
TRIANG
GLOBAL
((NIL) ((*1) G0012 G0010) ((*8) G0014) ((*3) G0017 G0015) ((*19) G0023 G0019 G0018) ((*2) G0025 G0024 G0023 G0022 G0021)
 ((*20) G0027 G0026 G0024 G0021) ((*18) G0032 G0030 G0027 G0026 G0025 G0022 G0020 G0019) ((*15) G0033 G0031 G0020 G0018)
 ((*14) G0016 G0015) ((*4) G0033 G0032 G0031 G0030) ((*11) G0037 G0035 G0034 G0029 G0016) ((*10) G0038 G0037 G0036 G0035
) ((*12) G0038 G0036 G0034 G0029 G0017) ((*16) G0028) ((*6) G0041 G0040) ((*5) G0046 G0045 G0042 G0041 G0040 G0039) ((*9
) G0046 G0044 G0039 G0014 G0013) ((*7) G0045 G0044 G0042 G0013 G0012 G0010) ((*17) G0047 G0043) ((*13) G0047 G0043 G0028
) ((*21)))
((NIL) (NIL) ((*8) G0014) ((*3) G0017 G0015) ((*19) G0023 G0019 G0018) (NIL) (NIL) (NIL) (NIL) ((*14) G0016 G0015) ((*15
 *2 *20 *18 *4) G0031 G0018 G0023 G0024 G0021 G0027 G0026 G0025 G0022 G0020 G0019 G0033 G0032 G0031 G0030) (NIL) (NIL) (
(*11 *10 *12) G0034 G0016 G0037 G0035 G0038 G0036 G0034 G0029 G0017) ((*16) G0028) (NIL) (NIL) (NIL) ((*6 *5 *9 *1 *7) G
0045 G0041 G0040 G0046 G0039 G0014 G0010 G0045 G0044 G0042 G0013 G0012 G0010) (NIL) ((*17 *13) G0043 G0047 G0043 G0028)
((*21)))
((NIL) ((*8) G0014) ((*3) G0017 G0015) (NIL) (NIL) (NIL) ((*14) G0016 G0015) ((*19 *15 *2 *20 *18 *4) G0019 G0031 G0018
G0023 G0024 G0021 G0027 G0026 G0025 G0022 G0020 G0019 G0033 G0032 G0031 G0030) (NIL) ((*11 *10 *12) G0034 G0016 G0037 G0
035 G0038 G0036 G0034 G0029 G0017) ((*16) G0028) (NIL) ((*6 *5 *9 *1 *7) G0045 G0041 G0040 G0046 G0039 G0014 G0010 G0045
 G0044 G0042 G0013 G0012 G0010) ((*17 *13) G0043 G0047 G0043 G0028) ((*21)))
LOCAL
(LOCAL ASSUMES (*3) (*14) SAME BODY)
(LOCAL ASSUMES (*8) (*6 *5 *9 *1 *7) SAME BODY)
((NIL) ((*6 *5 *9 *1 *7 *8) G0041 G0040 G0046 G0039 G0010 G0045 G0044 G0042 G0013 G0012 G0010 G0014) ((*14 *3) G0016 G00
17 G0015) (NIL) (NIL) ((*19 *15 *2 *20 *18 *4) G0019 G0031 G0018 G0023 G0024 G0021 G0027 G0026 G0025 G0022 G0020 G0019 G
0033 G0032 G0031 G0030) ((*11 *10 *12) G0034 G0016 G0037 G0035 G0038 G0036 G0034 G0029 G0017) ((*16) G0028) (NIL) ((*17
*13) G0043 G0047 G0043 G0028) ((*21)))
((NIL) ((*6 *5 *9 *1 *7 *8) G0041 G0040 G0046 G0039 G0010 G0045 G0044 G0042 G0013 G0012 G0010 G0014) (NIL) (NIL) ((*19 *
15 *2 *20 *18 *4) G0019 G0031 G0018 G0023 G0024 G0021 G0027 G0026 G0025 G0022 G0020 G0019 G0033 G0032 G0031 G0030) ((*14
 *3 *11 *10 *12) G0017 G0015 G0016 G0037 G0035 G0038 G0036 G0034 G0029 G0017) ((*16) G0028) ((*17 *13) G0043 G0047 G0043
 G0028) ((*21)))
LOCAL
(SINGLEBODY ASSUMES (*17 *13) (*16) SAME BODY)
(((*17 *13 *16) G0043 G0047 G0043 G0028) (NIL) ((*14 *3 *11 *10 *12) G0017 G0015 G0016 G0037 G0035 G0038 G0036 G0034 G00
29 G0017) ((*19 *15 *2 *20 *18 *4) G0019 G0031 G0018 G0023 G0024 G0021 G0027 G0026 G0025 G0022 G0020 G0019 G0033 G0032 G
0031 G0030) ((*6 *5 *9 *1 *7 *8) G0041 G0040 G0046 G0039 G0010 G0045 G0044 G0042 G0013 G0012 G0010 G0014))
LOCAL
SMB
RESULTS
(BODY 1. IS *17 *13 *16)
(BODY 2. IS *14 *3 *11 *10 *12)
(BODY 3. IS *19 *15 *2 *20 *18 *4)
(BODY 4. IS *6 *5 *9 *1 *7 *8)
NIL
```

BETTER RESULTS FOR L9

<u>Scenes R9 and R9T</u> Four bodies are found in R9, five in R9T. The difference is that Y and JA are not "matching T's" in R9T. The strong links among :12, :3, :10, and :16 are:



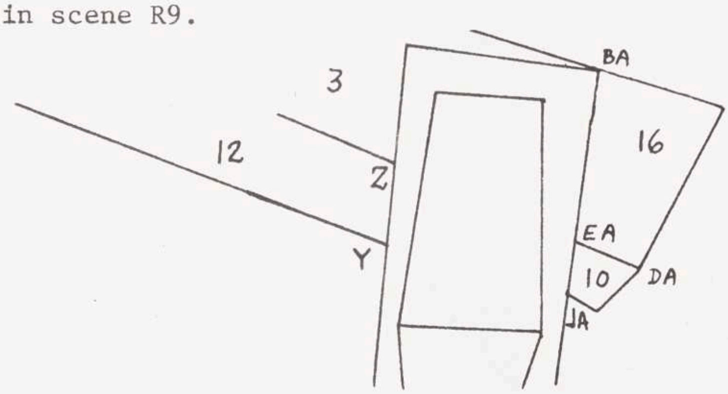
LINKS FOR R 9



LINKS FOR R 9 T

In R9, the two strong links (G0030 and G0021) between :12 and :10 were put by the matching T's Z-EA and Y-JA; of the two strong links between :10 and :16, one was because DA is an arrow; the other, because EA is a "T" for which heuristic (g) of table 'GLOBAL EVIDENCE' applies.

But in scene R9T, not being matching T's Y and JA, a link between :10 and :12 dissappears; and also nuclei :16 and :10 can not be linked by heuristic (g) of table 'GLOBAL EVIDENCE'. SEE decides to report two bodies there: :3-12 and :16-10 instead of one as in scene R9.



Are Y and JA matching T's or not? Different answers produce different analysis of the scene.

These scenes show that the analysis can be quite sensitive to the "right" definition of parallelism and colinearity.

```
SEE 58 ANALYZES R9
EVIDENCE
LOCALEVIDENCE
TRIANG
GLOBAL
((NIL) ((X:20) G0018 G0017 G0016 G0014 G0013 G0012) ((X:19) G0020 G0019 G0018 G0016 G0015) ((X:13) G0023 G0022 G0020) ((
X:15) G0026 G0025 G0023 G0022) ((X:10) G0031 G0030 G0028 G0021) ((X:16) G0031 G0029 G0028) ((X:8) G0027 G0019 G0015 G001
4 G0012) ((X:17) G0032 G0027 G0026 G0025 G0024) ((X:7) G0032 G0024 G0017 G0013) ((X:9) G0036 G0034 G0033) ((X:14) G0036
G0035 G0033) ((X:6) G0038 G0037) ((X:18) G0037) ((X:5) G0035 G0034) ((X:11) G0044 G0042 G0041 G0039) ((X:1) G0043 G0042
G0038) ((X:12) G0045 G0030 G0021) ((X:3) G0045 G0029) ((X:2) G0046 G0044 G0043 G0040 G0039) ((X:4) G0046 G0041 G0040) ((
X:21)))
((NIL) (NIL) (NIL) (NIL) (NIL) (NIL) (NIL) (NIL) (NIL) ((X:20 X:19 X:8 X:13 X:15 X:17 X:7) G0013 G0018 G0016 G0019 G0015
 G0014 G0012 G0020 G0023 G0022 G0027 G0026 G0025 G0032 G0024 G0017 G0013) (NIL) (NIL) ((X:6) G0038 G0037) ((X:18) G0037)
 ((X:9 X:14 X:5) G0036 G0033 G0035 G0034) (NIL) (NIL) (NIL) ((X:10 X:16 X:12 X:3) G0031 G0028 G0030 G0021 G0045 G0029) (
NIL) ((X:1 X:11 X:2 X:4) G0038 G0042 G0046 G0044 G0043 G0039 G0046 G0041 G0040) ((X:21)))
LOCAL
(LOCAL ASSUMES (X:6) (X:1 X:11 X:2 X:4) SAME BODY)
((NIL) (NIL) (NIL) (NIL) (NIL) ((X:20 X:19 X:8 X:13 X:15 X:17 X:7) G0013 G0018 G0016 G0019 G0015 G0014 G0012 G0020 G0023
 G0022 G0027 G0026 G0025 G0032 G0024 G0017 G0013) (NIL) ((X:1 X:11 X:2 X:4 X:6) G0042 G0044 G0043 G0039 G0046 G0041 G004
0 G0038 G0037) ((X:18) G0037) ((X:9 X:14 X:5) G0036 G0033 G0035 G0034) (NIL) ((X:10 X:16 X:12 X:3) G0031 G0028 G0030 G00
21 G0045 G0029) (NIL) ((X:21)))
LOCAL
(SINGLEBODY ASSUMES (X:1 X:11 X:2 X:4 X:6) (X:18) SAME BODY)
(((X:10 X:16 X:12 X:3) G0031 G0028 G0030 G0021 G0045 G0029) ((X:9 X:14 X:5) G0036 G0033 G0035 G0034) (NIL) ((X:1 X:11 X:
2 X:4 X:6 X:18) G0042 G0044 G0043 G0039 G0046 G0041 G0040 G0038 G0037) ((X:20 X:19 X:8 X:13 X:15 X:17 X:7) G0013 G0018 G
0016 G0019 G0015 G0014 G0012 G0020 G0023 G0022 G0027 G0026 G0025 G0032 G0024 G0017 G0013))
LOCAL
SMB
RESULTS
(BODY 1. IS X:10 X:16 X:12 X:3)
(BODY 2. IS X:9 X:14 X:5)
(BODY 3. IS X:1 X:11 X:2 X:4 X:6 X:18)
(BODY 4. IS X:20 X:19 X:8 X:13 X:15 X:17 X:7)
NIL
```

RESULTS FOR R9

R 9

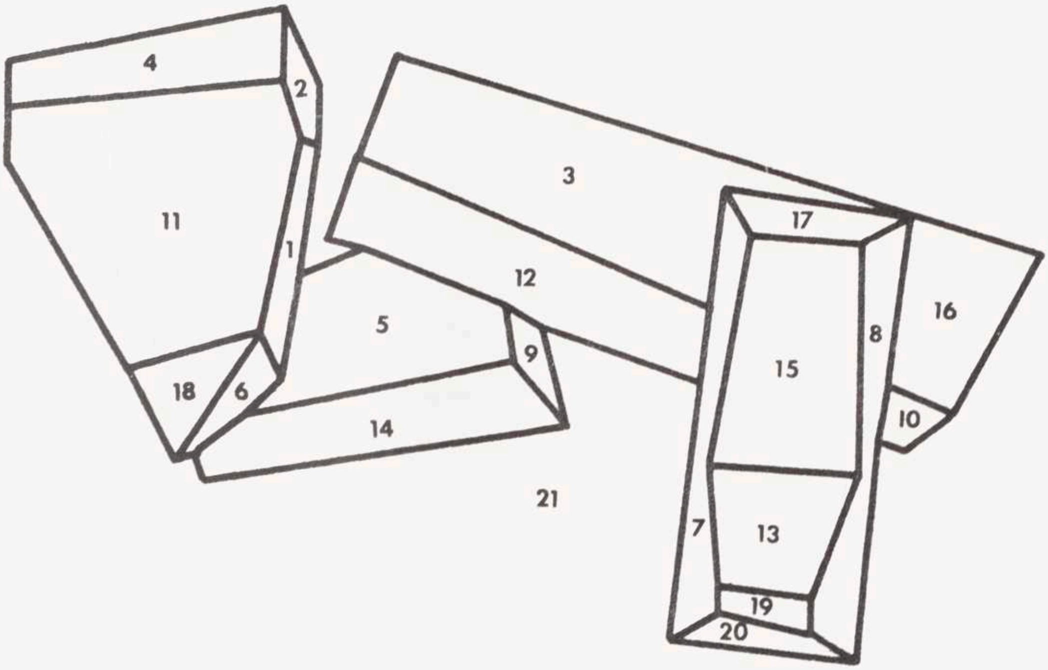


FIGURE 'R 9'

The four bodies were found.
SINGLEBODIES was needed to join :18
with :6-11-1-4-2.

R 9 T

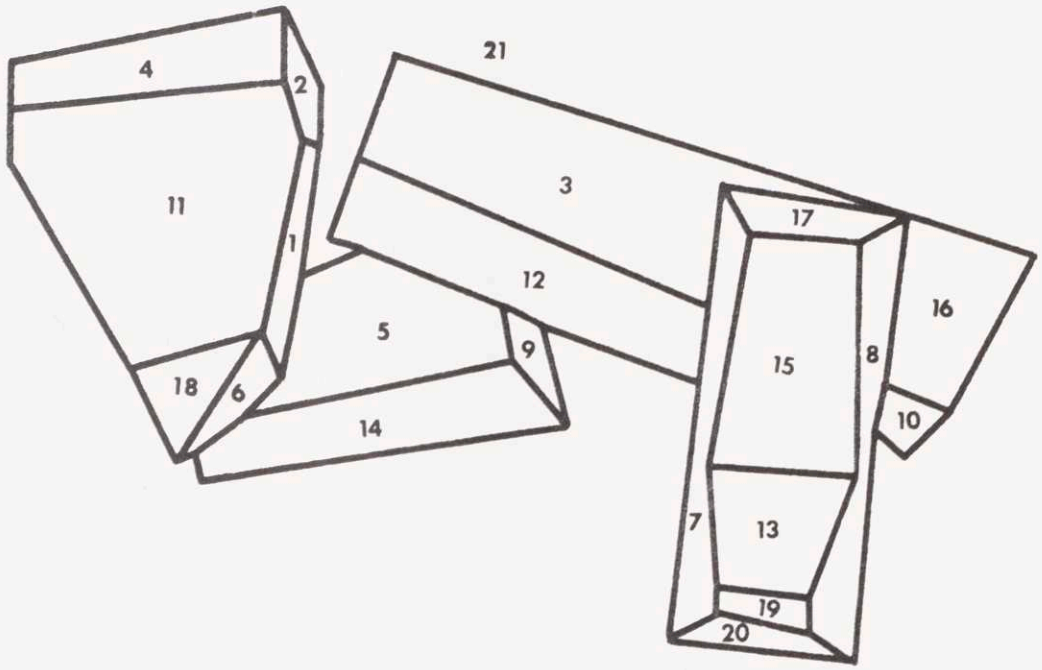


FIGURE 'R 9 T'

SINGLEBODIES joins :18 with the other portion of that body; LOCAL is needed to join :6 to that portion, and :16 with :10. Nevertheless, since :12 and :10 were not found to be the same face, body :16-10 is found, and body :12-3.

```
SEE 58 ANALYZES R9T
EVIDENCE
LOCALEVIDENCE
TRIANG
GLOBAL
((NIL) ((X:20) G0019 G0018 G0017 G0015 G0014 G0013) ((X:19) G0021 G0020 G0019 G0017 G0016) ((X:13) G0023 G0022 G0021) ((
X:15) G0026 G0025 G0023 G0022) ((X:10) G0030 G0029) ((X:16) G0030 G0028) ((X:8) G0027 G0020 G0016 G0015 G0013) ((X:17) G
0031 G0027 G0026 G0025 G0024) ((X:7) G0031 G0024 G0018 G0014) ((X:9) G0035 G0033 G0032) ((X:14) G0035 G0034 G0032) ((X:6
) G0037 G0036) ((X:18) G0036) ((X:5) G0034 G0033) ((X:11) G0043 G0041 G0040 G0038) ((X:1) G0042 G0041 G0037) ((X:12) G00
44 G0029) ((X:3) G0044 G0028) ((X:2) G0045 G0043 G0042 G0039 G0038) ((X:4) G0045 G0040 G0039) ((X:21)))
((NIL) (NIL) (NIL) (NIL) (NIL) ((X:10) G0030 G0029) ((X:16) G0030 G0028) (NIL) (NIL) ((X:20 X:19 X:8 X:13 X:15 X:17 X:7)
 G0014 G0019 G0017 G0020 G0016 G0015 G0013 G0021 G0023 G0022 G0027 G0026 G0025 G0031 G0024 G0018 G0014) (NIL) (NIL) ((X:
6) G0037 G0036) ((X:18) G0036) ((X:9 X:14 X:5) G0035 G0032 G0034 G0033) (NIL) (NIL) ((X:12) G0044 G0029) ((X:3) G0044 G0
028) (NIL) ((X:1 X:11 X:2 X:4) G0037 G0041 G0045 G0043 G0042 G0038 G0045 G0040 G0039) ((X:21)))
LOCAL
(LOCAL ASSUMES (X:6) (X:1 X:11 X:2 X:4) SAME BODY)
(LOCAL ASSUMES (X:10) (X:16) SAME BODY)
((NIL) (NIL) (NIL) ((X:16 X:10) G0028 G0030 G0029) (NIL) (NIL) ((X:20 X:19 X:8 X:13 X:15 X:17 X:7) G0014 G0019 G0017 G00
20 G0016 G0015 G0013 G0021 G0023 G0022 G0027 G0026 G0025 G0031 G0024 G0018 G0014) (NIL) ((X:1 X:11 X:2 X:4 X:6) G0041 G0
043 G0042 G0038 G0045 G0040 G0039 G0037 G0036) ((X:18) G0036) ((X:9 X:14 X:5) G0035 G0032 G0034 G0033) (NIL) ((X:12) G00
44 G0029) ((X:3) G0044 G0028) (NIL) ((X:21)))
LOCAL
(SINGLEBODY ASSUMES (X:1 X:11 X:2 X:4 X:6) (X:18) SAME BODY)
(((X:3) G0044 G0028) ((X:12) G0044 G0029) ((X:9 X:14 X:5) G0035 G0032 G0034 G0033) (NIL) ((X:1 X:11 X:2 X:4 X:6 X:18) G0
041 G0043 G0042 G0038 G0045 G0040 G0039 G0037 G0036) ((X:20 X:19 X:8 X:13 X:15 X:17 X:7) G0014 G0019 G0017 G0020 G0016 G
0015 G0013 G0021 G0023 G0022 G0027 G0026 G0025 G0031 G0024 G0018 G0014) ((X:16 X:10) G0028 G0030 G0029))
LOCAL
SMB
(SMB ASSUMES X:3 X:12 SAME BODY)
RESULTS
(BODY 1. IS X:3 X:12)
(BODY 2. IS X:9 X:14 X:5)
(BODY 3. IS X:1 X:11 X:2 X:4 X:6 X:18)
(BODY 4. IS X:20 X:19 X:8 X:13 X:15 X:17 X:7)
(BODY 5. IS X:16 X:10)
NIL
```

RESULTS FOR R9T

<u>Scene TRIAL</u> This scene has been analyzed in great detail in the section that describes the program SEE. Its links are found in graphic form in figure 'TRIAL - LINKS', or in written form (lists) in "RESULTS FOR TRIAL".

LOCAL had to join :13 with the remainder of that body.

TRIAL

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

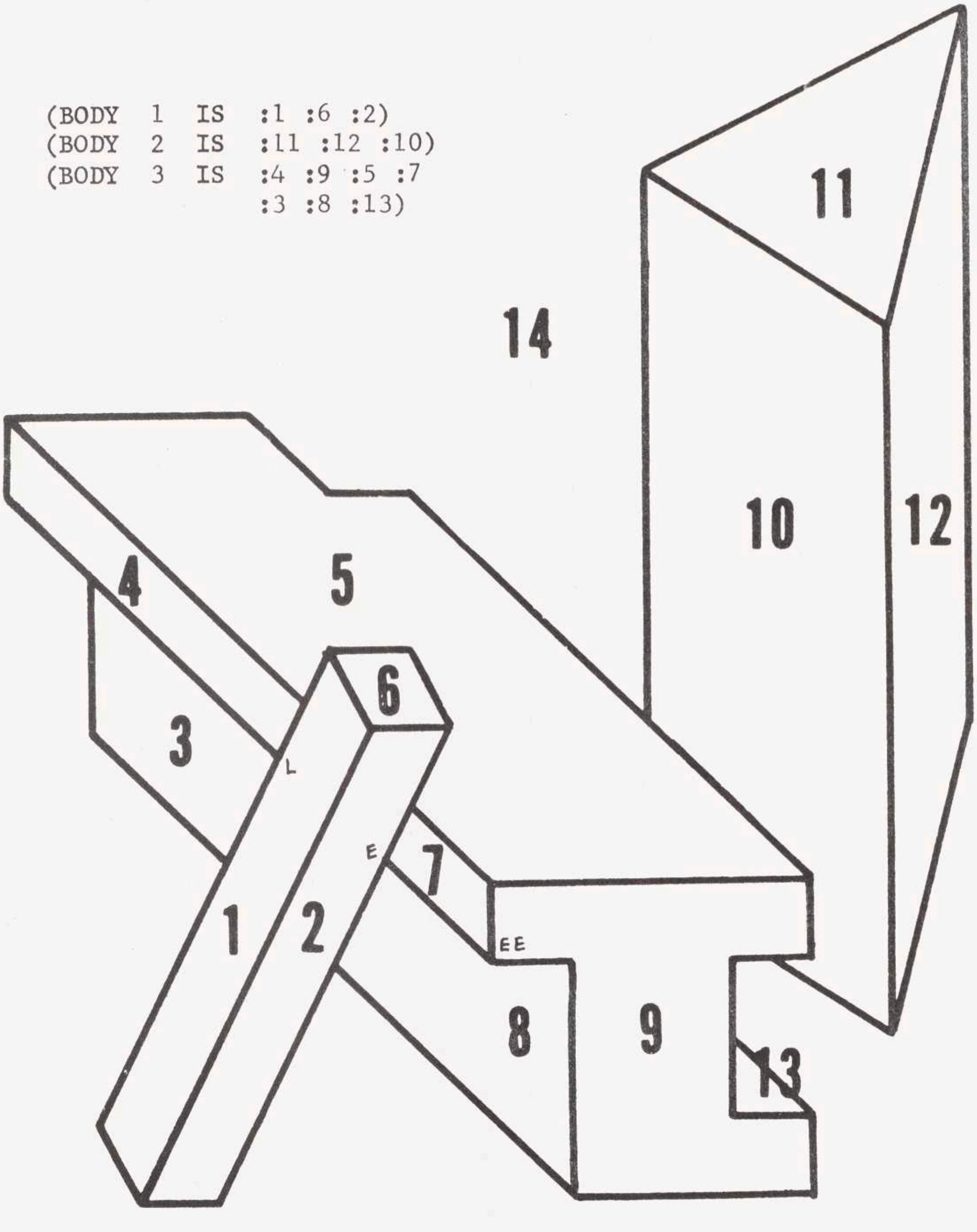


FIGURE 'T R I A L'

```
SEE 58 ANALYZES TRIAL
EVIDENCE
LOCALEVIDENCE
TRIANG
GLOBAL
((NIL) ((*11) G0014 G0013 G0011 G0010) ((*12) G0015 G0014 G0013 G0012) ((*13) G0021) ((*9) G0022 G0021 G0020 G0019 G0017
 G0016) ((*10) G0015 G0012 G0011 G0010) ((*3) G0034 G0025 G0024) ((*4) G0033 G0032 G0026 G0025 G0023) ((*6) G0031 G0030
G0029 G0027) ((*5) G0026 G0023 G0022 G0018 G0017) ((*7) G0033 G0032 G0019 G0018 G0016) ((*8) G0034 G0024 G0020) ((*2) G0
035 G0031 G0029 G0028) ((*14)) ((*1) G0035 G0030 G0028 G0027))
((NIL) (NIL) (NIL) ((*13) G0021) (NIL) ((*11 *12 *10) G0011 G0015 G0014 G0013 G0015 G0012 G0011 G0010) (NIL) (NIL) (NIL)
 (NIL) (NIL) ((*4 *9 *5 *7 *3 *8) G0025 G0021 G0016 G0026 G0023 G0022 G0017 G0033 G0032 G0019 G0018 G0016 G0025 G0034 G0
024 G0020) (NIL) ((*14)) ((*6 *2 *1) G0027 G0031 G0029 G0035 G0030 G0028 G0027))
LOCAL
(LOCAL ASSUMES (*13) (*4 *9 *5 *7 *3 *8) SAME BODY)
LOCAL
((NIL) (NIL) ((*4 *9 *5 *7 *3 *8 *13) G0021 G0026 G0023 G0022 G0017 G0033 G0032 G0019 G0018 G0016 G0025 G0034 G0024 G002
0 G0021) ((*11 *12 *10) G0011 G0015 G0014 G0013 G0015 G0012 G0011 G0010) (NIL) (NIL) (NIL) ((*14)) ((*6 *2 *1) G0027 G00
31 G0029 G0035 G0030 G0028 G0027))
LOCAL
(((*6 *2 *1) G0027 G0031 G0029 G0035 G0030 G0028 G0027) ((*11 *12 *10) G0011 G0015 G0014 G0013 G0015 G0012 G0011 G0010)
((*4 *9 *5 *7 *3 *8 *13) G0021 G0026 G0023 G0022 G0017 G0033 G0032 G0019 G0018 G0016 G0025 G0034 G0024 G0020 G0021))
LOCAL
SME
RESULTS
(BODY 1. IS *6 *2 *1)
(BODY 2. IS *11 *12 *10)
(BODY 3. IS *4 *9 *5 *7 *3 *8 *13)
NIL
```

RESULTS FOR TRIAL

<u>Scene ARCH</u> SEE analyzes scene ARCH (see figure 'ARCH') with results displayed in 'RESULTS FOR ARCH'. This is an scene composed by many degenerate views of objects. It is an ambiguous scene (see section on Optical Illusions), in that several good interpretations are possible.

The program reports :7 and :17 as one body, which could be plausible. :16, :9 and :10 get reported as independent objects. In the scene from where this picture or line drawing was taken, :7, :17 and :16 were the vertical face of an object. :10 was the vertical face of another, :9 being its horizontal (top) face. In cases like this, in order to choose the "right" one of several possible interpretations, more information has to be supplied to the program, such as lighting, textures, color, etc.

No link was put by A between :3 and :29, or by UB between :5 and :19, because D and W are GOODTs. In one case, G provides with more links and causes :3-8-29-31 to be reported as one body, which is correct; in the other case, Q can not supply any links, and that body is split in two: :5-4 and :19-18. This is a mistake of GOODT, who accepts W as a genuine T. If this were not the case, the arrow UB would establish a link between :5 and :19, avoiding the mistake. GOODT could stand some improvement.

The body :22-23 was identified correctly.

ARCH

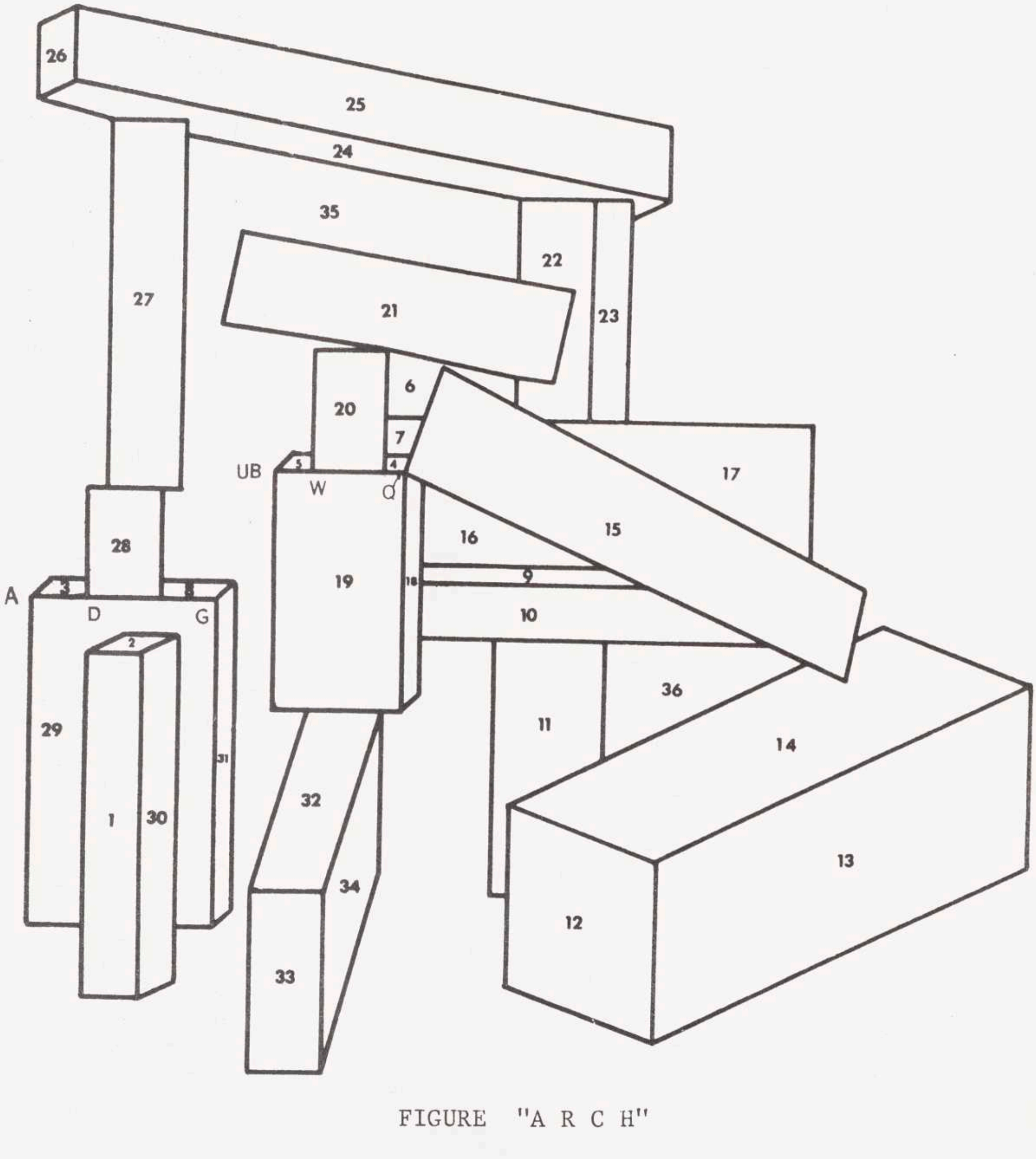


FIGURE "A R C H"

Ambiguous scene that could be correctly interpreted in several different manners. :7-17 was reported as a single body (see table 'RESULTS FOR ARCH'), and also :9.

The body :5-4-19-18 was split in two: :5-4 and :19-18, but not :3-8-29-31, which was counted as one body.

```
SEE 58 ANALYZES ARCH
EVIDENCE
LOCALEVIDENCE
TRIANG
GLOBAL
((NIL) ((:3) G0022) ((:8) G0025 G0023 G0022) ((:31) G0029 G0025 G0024 G0023) ((:4) G0032) ((:26) G0038 G0037 G0035 G0034
) ((:24) G0039 G0036 G0035 G0034) ((:25) G0039 G0038 G0037 G0036) ((:23) G0040) ((:17) G0033) ((:13) G0045 G0044 G0042 G
0041) ((:12) G0046 G0044 G0043 G0042) ((:11)) ((:36)) ((:14) G0046 G0045 G0043 G0041) ((:9)) ((:16)) ((:15)) ((:10)) ((:
33) G0050 G0049 G0047 G0030) ((:34) G0050 G0048 G0047) ((:32) G0049 G0048 G0030) ((:5) G0032) ((:19) G0031) ((:7) G0033)
 ((:6)) ((:20)) ((:22) G0040) ((:21)) ((:27)) ((:28)) ((:2) G0052 G0051 G0027 G0026) ((:1) G0053 G0052 G0028 G0027) ((:3
0) G0053 G0051 G0028 G0026) ((:29) G0029 G0024) ((:35)) ((:18) G0031))
((NIL) ((:3) G0022) (NIL) (NIL) ((:4) G0032) (NIL) (NIL) ((:26 :24 :25) G0037 G0039 G0035 G0034 G0039 G0038 G0037 G0036)
 ((:23) G0040) ((:17) G0033) (NIL) (NIL) ((:11)) ((:36)) ((:13 :12 :14) G0042 G0046 G0044 G0042 G0046 G0045 G0043 G0041)
 ((:9)) ((:16)) ((:15)) ((:10)) (NIL) (NIL) ((:33 :34 :32) G0030 G0050 G0047 G0049 G0048 G0030) ((:5) G0032) ((:19) G003
1) ((:7) G0033) ((:6)) ((:20)) ((:22) G0040) ((:21)) ((:27)) ((:28)) (NIL) (NIL) ((:2 :1 :30) G0026 G0052 G0027 G0053 G0
051 G0028 G0026) ((:8 :31 :29) G0022 G0025 G0023 G0029 G0024) ((:35)) ((:18) G0031))
LOCAL
(LOCAL ASSUMES (:19) (:18) SAME BODY)
(LOCAL ASSUMES (:3) (:8 :31 :29) SAME BODY)
LOCAL
((NIL) ((:8 :31 :29 :3) G0025 G0023 G0029 G0024 G0022) (NIL) ((:4) G0032) (NIL) ((:26 :24 :25) G0037 G0039 G0035 G0034 G
0039 G0038 G0037 G0036) ((:23) G0040) ((:17) G0033) (NIL) ((:11)) ((:36)) ((:13 :12 :14) G0042 G0046 G0044 G0042 G0046 G
0045 G0043 G0041) ((:9)) ((:16)) ((:15)) ((:10)) (NIL) ((:33 :34 :32) G0030 G0050 G0047 G0049 G0048 G0030) ((:5) G0032)
((:18 :19) G0031) ((:7) G0033) ((:6)) ((:20)) ((:22) G0040) ((:21)) ((:27)) ((:28)) (NIL) ((:2 :1 :30) G0026 G0052 G0027
 G0053 G0051 G0028 G0026) (NIL) ((:35)) (NIL))
LOCAL
(SINGLEBODY ASSUMES (:23) (:22) SAME BODY)
(SINGLEBODY ASSUMES (:17) (:7) SAME BODY)
(SINGLEBODY ASSUMES (:4) (:5) SAME BODY)
(((:2 :1 :30) G0026 G0052 G0027 G0053 G0051 G0028 G0026) ((:28)) ((:27)) ((:21)) (NIL) ((:20)) (NIL) ((:18 :19) G0031) (
NIL) ((:33 :34 :32) G0030 G0050 G0047 G0049 G0048 G0030) ((:10)) ((:15)) ((:16)) ((:9)) ((:13 :12 :14) G0042 G0046 G0044
 G0042 G0046 G0045 G0043 G0041) ((:11)) ((:17 :7) G0033) ((:23 :22) G0040) ((:26 :24 :25) G0037 G0039 G0035 G0034 G0039
G0038 G0037 G0036) ((:4 :5) G0032) ((:8 :31 :29 :3) G0025 G0023 G0029 G0024 G0022))
LOCAL
SMB
RESULTS
(THE FIRST 9. BODIES ARE ((:28)) ((:27)) ((:21)) ((:20)) ((:10)) ((:15)) ((:16)) ((:9)) ((:11)))
(BODY 10. IS :2 :1 :30)
(BODY 11. IS :18 :19)
(BODY 12. IS :33 :34 :32)
(BODY 13. IS :13 :12 :14)
(BODY 14. IS :17 :7)
(BODY 15. IS :23 :22)
(BODY 16. IS :26 :24 :25)
(BODY 17. IS :4 :5)
(BODY 18. IS :8 :31 :29 :3)
NIL
```

RESULTS FOR ARCH

Scene HARD This scene consists of objects of the same shape, namely triangular prisms. All are correctly identified, including the long and twice occluded :3-21-22-23-24-28-29. :1-22-33 was also found. LOCAL had to be used to join :15 with :16, and also :11 with :12.

In an older version of the program, :7 was identified as a single body, and :6 as another, because they have no visible "useful" vertices to place links {Guzmán PISA 68}. Now SEE joins :6 and :7, because both are "GOODPALs". See "Operation of the Program; SMB"(page 99).

These scenes are sometimes obtained from a picture, so that they are the result of a perspective transformation. Some other scenes are drawn more or less in an orthogonal or isometric projection. SEE does not depend heavily in the type of projection; there are only a few heuristics that use notions of parallelism.

HARD

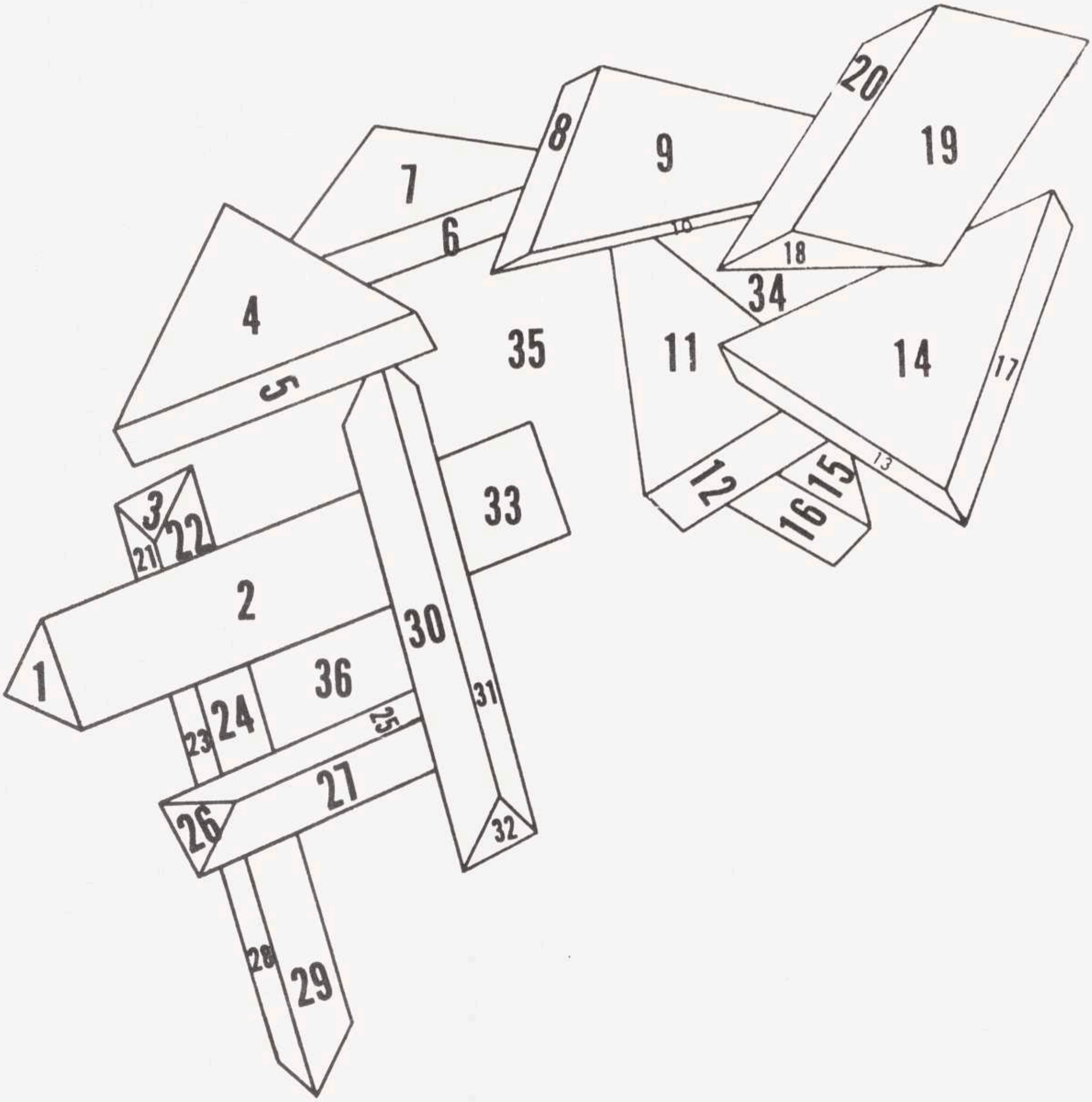


FIGURE 'H A R D'

All the bodies were correctly found.
The most difficult was :6-7, since SMB
had to join both regions, which do
not have "useful" visible vertices.

```
SEE 58 ANALYZES HARD
EVIDENCE
LOCALEVIDENCE
TRIANG
GLOBAL
((NIL) ((:34)) ((:6)) ((:36)) ((:24) G0026 G0025 G0023 G0021 G0020) ((:23) G0028 G0027 G0024 G0022 G0021) ((:13) G0046 G
0044 G0043 G0042) ((:17) G0047 G0046 G0045 G0044) ((:7)) ((:22) G0049 G0041 G0040 G0029 G0026 G0025) ((:3) G0050 G0049 G
0041 G0039) ((:21) G0050 G0040 G0039 G0029 G0028 G0027) ((:1) G0052 G0051) ((:2) G0052 G0051 G0018 G0017) ((:25) G0053 G
0038 G0036 G0019) ((:26) G0054 G0053 G0037 G0036) ((:27) G0054 G0038 G0037 G0019) ((:28) G0055 G0024 G0022 G0015) ((:29)
 G0055 G0023 G0020 G0015) ((:32) G0057 G0056 G0034 G0033) ((:33) G0018 G0017) ((:31) G0057 G0035 G0034 G0016) ((:5) G005
8 G0048) ((:4) G0058 G0048) ((:10) G0059 G0032 G0031) ((:14) G0047 G0045 G0043 G0042) ((:18) G0063 G0061 G0060 G0014) ((
:19) G0064 G0063 G0062 G0061) ((:20) G0064 G0062 G0060 G0014) ((:9) G0065 G0032 G0030) ((:8) G0065 G0059 G0031 G0030) ((
:30) G0056 G0035 G0033 G0016) ((:15) G0066) ((:16) G0066) ((:35)) ((:11) G0067) ((:12) G0067))
((NIL) ((:34)) ((:6)) ((:36)) (NIL) (NIL) (NIL) (NIL) ((:7)) (NIL) (NIL) (NIL) (NIL) (NIL) (NIL) (NIL) ((:25 :26 :27) G0
019 G0053 G0036 G0054 G0038 G0037 G0019) (NIL) ((:24 :22 :3 :23 :21 :28 :29) G0020 G0026 G0025 G0049 G0041 G0021 G0050 G
0040 G0039 G0029 G0028 G0027 G0024 G0022 G0055 G0023 G0020 G0015) (NIL) ((:1 :2 :33) G0052 G0051 G0017 G0018 G0017) (NIL
) (NIL) ((:5 :4) G0048 G0058 G0048) (NIL) ((:13 :17 :14) G0043 G0047 G0046 G0044 G0047 G0045 G0043 G0042) (NIL) (NIL) ((
:18 :19 :20) G0060 G0064 G0063 G0061 G0064 G0062 G0060 G0014) (NIL) ((:9 :10 :8) G0032 G0032 G0065 G0059 G0031 G0030) ((
:32 :31 :30) G0033 G0057 G0034 G0056 G0035 G0033 G0016) ((:15) G0066) ((:16) G0066) ((:35)) ((:11) G0067) ((:12) G0067))

LOCAL
(LOCAL ASSUMES (:11) (:12) SAME BODY)
(LOCAL ASSUMES (:15) (:16) SAME BODY)
((NIL) ((:34)) ((:6)) ((:36)) (NIL) (NIL) ((:7)) (NIL) (NIL) (NIL) ((:25 :26 :27) G0019 G0053 G0036 G0054 G0038 G0037 G0
019) ((:24 :22 :3 :23 :21 :28 :29) G0020 G0026 G0025 G0049 G0041 G0021 G0050 G0040 G0039 G0029 G0028 G0027 G0024 G0022 G
0055 G0023 G0020 G0015) ((:1 :2 :33) G0052 G0051 G0017 G0018 G0017) (NIL) ((:5 :4) G0048 G0058 G0048) ((:13 :17 :14) G00
43 G0047 G0046 G0044 G0047 G0045 G0043 G0042) (NIL) ((:18 :19 :20) G0060 G0064 G0063 G0061 G0064 G0062 G0060 G0014) ((:9
 :10 :8) G0032 G0032 G0065 G0059 G0031 G0030) ((:32 :31 :30) G0033 G0057 G0034 G0056 G0035 G0033 G0016) ((:16 :15) G0066
) (NIL) ((:35)) ((:12 :11) G0067) (NIL))
LOCAL
(((:12 :11) G0067) ((:16 :15) G0066) ((:32 :31 :30) G0033 G0057 G0034 G0056 G0035 G0033 G0016) ((:9 :10 :8) G0032 G0032
G0065 G0059 G0031 G0030) ((:18 :19 :20) G0060 G0064 G0063 G0061 G0064 G0062 G0060 G0014) ((:13 :17 :14) G0043 G0047 G004
6 G0044 G0047 G0045 G0043 G0042) ((:5 :4) G0048 G0058 G0048) ((:1 :2 :33) G0052 G0051 G0017 G0018 G0017) ((:24 :22 :3 :2
3 :21 :28 :29) G0020 G0026 G0025 G0049 G0041 G0021 G0050 G0040 G0039 G0029 G0028 G0027 G0024 G0022 G0055 G0023 G0020 G00
15) ((:25 :26 :27) G0019 G0053 G0036 G0054 G0038 G0037 G0019) ((:7)) ((:6)))
LOCAL
SMB
(SMB ASSUMES :7 :6 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)
(BODY 7. IS :5 :4)
(BODY 8. IS :1 :2 :33)
(BODY 9. IS :24 :22 :3 :23 :21 :28 :29)
(BODY 10. IS :25 :26 :27)
(BODY 11. IS :7 :6)
NIL
```

RESULTS FOR HARD

Scene L4 The body :10-9 was reported isolated from :13-2-3, due to insufficiency of links. See comments to figure R17, also. The algorithm that localizes matching T's could stand improvement. It sometimes produces "bad links" such as between :4 and :13, and between :6 and :3, because it found two T's that looked like they were matching (this mistake did not happen, actually, because vertex R is not a T, but a fork!), EA and R in this case. The suggestion in page 173 will lessen, but not suppress, these "mistakes".

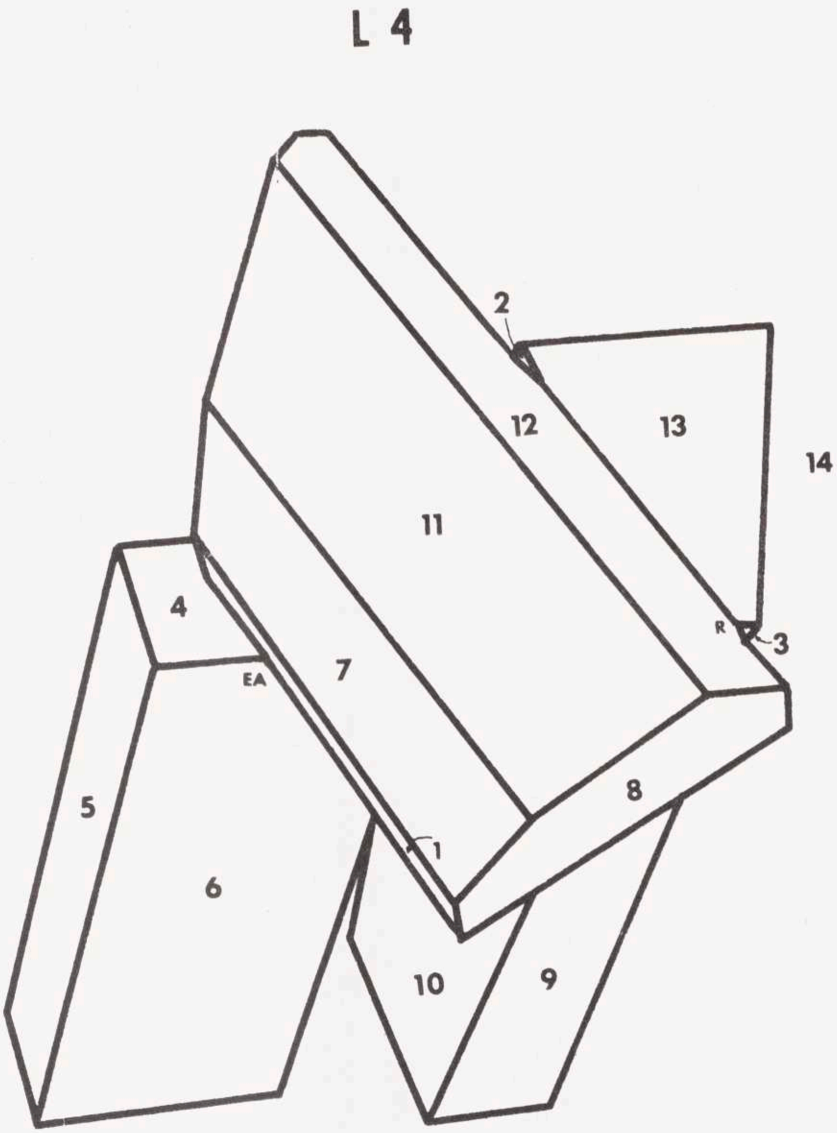


FIGURE 'L 4'

Body :2-3-13 was reported separated from body :10-9. Not enough T-joints.

```
SEE 58 ANALYZES L4
EVIDENCE
LOCALEVIDENCE
TRIANG
GLOBAL
((NIL) ((*2) G0013) ((*11) G0018 G0017 G0016 G0014 G0012 G0011) ((*13) G0021 G0020 G0013) ((*3) G0021 G0020) ((*12) G002
2 G0019 G0017 G0012) ((*9) G0023) ((*7) G0025 G0024 G0015 G0014 G0011) ((*8) G0027 G0026 G0024 G0022 G0019 G0018 G0016 G
0015) ((*1) G0027 G0026 G0025) ((*10) G0023) ((*4) G0031 G0029 G0028 G0010) ((*5) G0032 G0031 G0030 G0029) ((*6) G0032 G
0030 G0028 G0010) ((*14)))
((NIL) ((*2) G0013) (NIL) (NIL) ((*13 *3) G0020 G0013 G0021 G0020) (NIL) ((*9) G0023) (NIL) (NIL) ((*11 *12 *7 *8 *1) G0
022 G0017 G0012 G0015 G0014 G0011 G0026 G0024 G0022 G0019 G0018 G0016 G0015 G0027 G0026 G0025) ((*10) G0023) (NIL) (NIL)
 ((*4 *5 *6) G0028 G0032 G0031 G0029 G0032 G0030 G0028 G0010) ((*14)))
LOCAL
(LOCAL ASSUMES (*9) (*10) SAME BODY)
((NIL) ((*2) G0013) (NIL) ((*13 *3) G0020 G0013 G0021 G0020) ((*10 *9) G0023) (NIL) ((*11 *12 *7 *8 *1) G0022 G0017 G001
2 G0015 G0014 G0011 G0026 G0024 G0022 G0019 G0018 G0016 G0015 G0027 G0026 G0025) (NIL) (NIL) ((*4 *5 *6) G0028 G0032 G00
31 G0029 G0032 G0030 G0028 G0010) ((*14)))
LOCAL
(SINGLEBODY ASSUMES (*13 *3) (*2) SAME BODY)
(((*4 *5 *6) G0028 G0032 G0031 G0029 G0032 G0030 G0028 G0010) ((*11 *12 *7 *8 *1) G0022 G0017 G0012 G0015 G0014 G0011 G0
026 G0024 G0022 G0019 G0018 G0016 G0015 G0027 G0026 G0025) ((*10 *9) G0023) ((*13 *3 *2) G0020 G0013 G0021 G0020) (NIL))

LOCAL
SMB
RESULTS
(BODY 1. IS *4 *5 *6)
(BODY 2. IS *11 *12 *7 *8 *1)
(BODY 3. IS *10 *9)
(BODY 4. IS *13 *3 *2)
NIL
```

RESULTS FOR L4

Scene R4 The table 'RESULTS FOR R4' shows what happens when the tolerances are too large. Five bodies are found. Vertex B is considered to be a "T", and inhibits the links suggested by the Arrows R and A. As a result, :1 gets cut off :7-9-5-10.

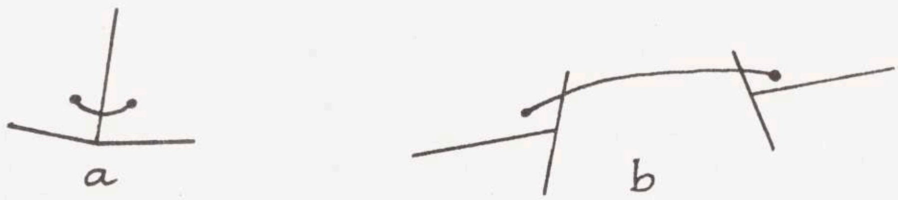
The way :2 gets isolated is as follows: T and AA claim to be matching T's, the link suggested by U is inhibited by Z (a Corner), and :2 gets disconnected from :3-4.

The correct solution is obtained after reducing the values of COLTO and SINTO to 0.05 and 0.005 (see listings; COLTO decides if two lines are colinear, SINTO if they are parallel), respectively. The results appear also in 'RESULTS FOR R4', and we can see now that only three bodies (the correct ones) are identified.

Suggestion Lines like the one below should be "straightened" either by SEE or (better) by the preprocessor; for example, B K L N and D G H O in figure R17. See section 'On Noisy Input'.

SUGGESTION

Conservatism and Tolerance More strict tolerances do not make the program more conservative in all cases: the link in (a) fails to be placed if the program has too loose (large) tolerances, because A will be transformed into a "T" (it will be considered to be a "T"), losing the link; the link in (b) fails to be laid if the tolerances are too strict, because the T-joints will not be colinear.



In (a), links disappear if tolerances are too big; in (b), if they are too small. In both cases, conservative behavior (cf. page 212) appears.

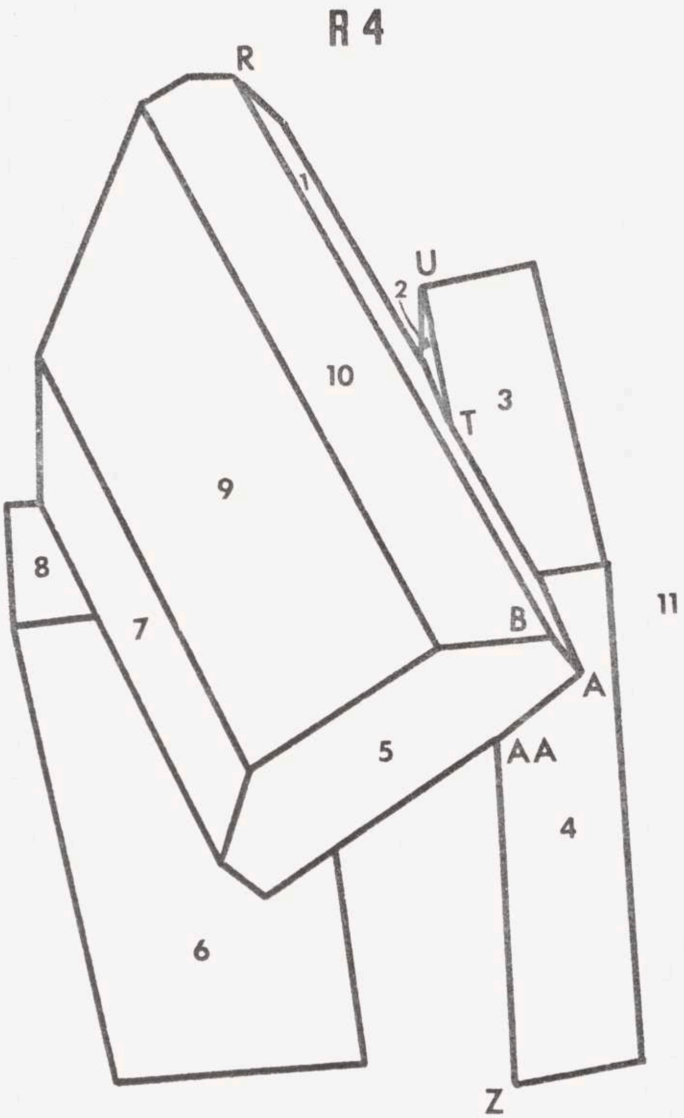


FIGURE 'R 4'

Either three or five bodies are found, according to the values of certain parameters. These scenes are "noisy" in the sense that the coordinates of the vertices depart from their "ideal" position by as much as one millimeter, or about 1 % of the total size of the image, which is about one decimeter. This error is not large enough to affect long lines, but it may substantially change the direction of short segments.

```
SEE 58 ANALYZES R4
EVIDENCE
LOCALEVIDENCE
TRIANG
GLOBAL
((NIL) ((X:9) G0017 G0015 G0013 G0012 G0011 G0010) ((X:7) G0018 G0016 G0015 G0010) ((X:6) G0022 G0020 G0019) ((X:8) G002
2 G0021 G0019) ((X:10) G0014 G0012 G0011) ((X:2)) ((X:3) G0023 G0021 G0009) ((X:11)) ((X:5) G0018 G0017 G0016 G0014 G001
3) ((X:1)) ((X:4) G0023 G0020 G0009))
((NIL) (NIL) (NIL) (NIL) ((X:6 X:8) G0020 G0022 G0021 G0019) (NIL) ((X:2)) (NIL) ((X:11)) ((X:9 X:7 X:10 X:5) G0013 G001
8 G0015 G0010 G0012 G0011 G0018 G0017 G0016 G0014 G0013) ((X:1)) ((X:3 X:4) G0021 G0023 G0020 G0009))
LOCAL
((NIL) (NIL) ((X:6 X:8) G0020 G0022 G0021 G0019) ((X:2)) ((X:11)) ((X:9 X:7 X:10 X:5) G0013 G0018 G0015 G0010 G0012 G001
1 G0018 G0017 G0016 G0014 G0013) ((X:1)) ((X:3 X:4) G0021 G0023 G0020 G0009))
LOCAL
(((X:3 X:4) G0021 G0023 G0020 G0009) ((X:1)) ((X:9 X:7 X:10 X:5) G0013 G0018 G0015 G0010 G0012 G0011 G0018 G0017 G0016 G
0014 G0013) ((X:2)) ((X:6 X:8) G0020 G0022 G0021 G0019))
LOCAL
SMB
RESULTS
(THE FIRST 2. BODIES ARE ((X:1)) ((X:2)))
(BODY 3. IS X:3 X:4)
(BODY 4. IS X:9 X:7 X:10 X:5)
(BODY 5. IS X:6 X:8)
NIL
```

RESULTS FOR R4

```
(CHANGES TO SINTO AND COLTO)
```

⇐

```
0.5E-1

0.5E-2

LLENA
FOOP
TYPEGENERATOR
MATES
NEXTE
105

SEE 58 ANALYZES R4
EVIDENCE
LOCALEVIDENCE
TRIANG
GLOBAL
((NIL) ((X:9) G0020 G0018 G0016 G0015 G0014 G0013) ((X:7) G0021 G0019 G0018 G0013) ((X:6) G0022) ((X:8) G0022) ((X:10) G
0023 G0017 G0015 G0014 G0011 G0009) ((X:2) G0025 G0024) ((X:3) G0026 G0025 G0024 G0012) ((X:11)) ((X:5) G0027 G0021 G002
0 G0019 G0017 G0016 G0010 G0009) ((X:1) G0027 G0023 G0011 G0010) ((X:4) G0026 G0012))
((NIL) (NIL) (NIL) ((X:6) G0022) ((X:8) G0022) (NIL) (NIL) (NIL) ((X:11)) (NIL) ((X:9 X:7 X:10 X:5 X:1) G0021 G0018 G001
3 G0015 G0014 G0027 G0021 G0020 G0019 G0017 G0016 G0009 G0027 G0023 G0011 G0010) ((X:2 X:3 X:4) G0026 G0025 G0024 G0026
G0012))
LOCAL
(LOCAL ASSUMES (X:6) (X:8) SAME BODY)
((NIL) (NIL) ((X:8 X:6) G0022) (NIL) (NIL) ((X:11)) ((X:9 X:7 X:10 X:5 X:1) G0021 G0018 G0013 G0015 G0014 G0027 G0021 G0
020 G0019 G0017 G0016 G0009 G0027 G0023 G0011 G0010) ((X:2 X:3 X:4) G0026 G0025 G0024 G0026 G0012))
LOCAL
(((X:2 X:3 X:4) G0026 G0025 G0024 G0026 G0012) ((X:9 X:7 X:10 X:5 X:1) G0021 G0018 G0013 G0015 G0014 G0027 G0021 G0020 G
0019 G0017 G0016 G0009 G0027 G0023 G0011 G0010) ((X:8 X:6) G0022))
LOCAL
SMB
RESULTS
(BODY 1. IS X:2 X:3 X:4)
(BODY 2. IS X:9 X:7 X:10 X:5 X:1)
(BODY 3. IS X:8 X:6)
NIL
```

RESULTS FOR R4

Scene MOMO The long body :29-30-34-20-19 gets identified as follows: :29 and :30 get two links, and :30 with :19 also, so we have the nucleus :29-30-19. Two links (because of matching T's) join :34 with :20, to form nucleus :34-20. Regions :30 and :34 receive a strong link, by heuristic (g) of table 'GLOBAL EVIDENCE', and :19 with :20 by the same reason. That completes the body.

The fork that is common to :12, 13 and 14 puts a link between :12 and :13, but it is not enough to cause mis-recognition. A link is put by that same Fork between :13 and :14, as it should be, but the link between :12 and :14 is inhibited by NOSABO.

There is a program that finds regions of a scene belonging to the background, when not indicated as such in the input. For MOMO, the results of this program appear in page 131 .

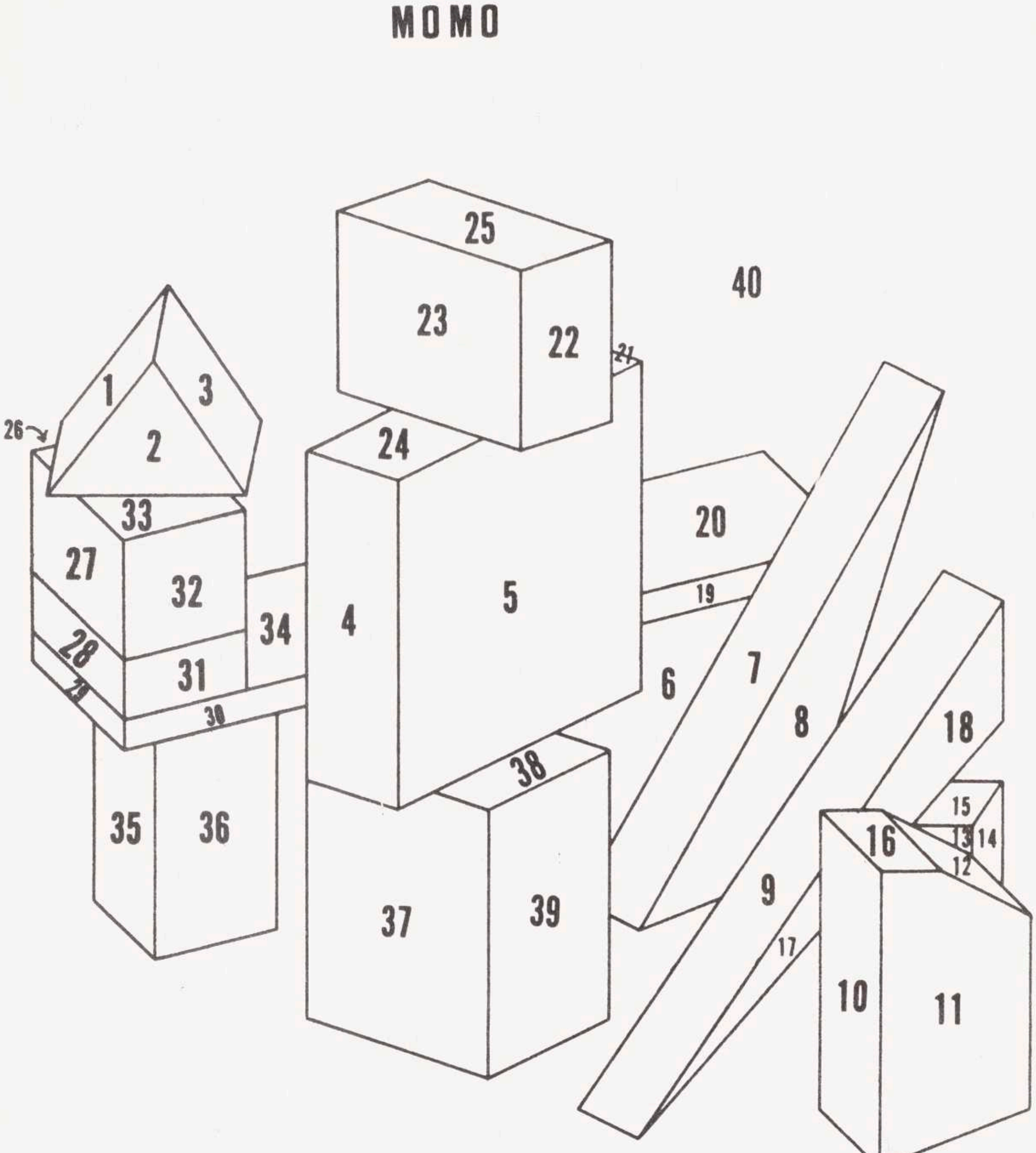


FIGURE 'M O M O'
All bodies are correctly identified.

```
SEE 58 ANALYZES MOMO
EVIDENCE
LOCALEVIDENCE
TRIANG
GLOBAL
((NIL) ((:38) G0044 G0043 G0041 G0040) ((:19) G0046 G0045 G0020) ((:6)) ((:20) G0050 G0046 G0021) ((:25) G0053 G0052 G00
32 G0031) ((:23) G0053 G0034 G0033 G0032) ((:24) G0075 G0054 G0037 G0036 G0035) ((:34) G0050 G0021 G0019) ((:37) G0057 G
0042 G0041 G0040) ((:39) G0057 G0044 G0043 G0042) ((:7) G0058 G0049) ((:8) G0058 G0049) ((:17) G0065 G0059) ((:10) G0063
 G0061 G0060 G0047) ((:9) G0059 G0048) ((:16) G0069 G0068 G0066 G0063 G0062 G0060) ((:11) G0069 G0067 G0064 G0062 G0061
G0047) ((:14) G0073 G0072 G0071 G0070) ((:13) G0074 G0072 G0071) ((:12) G0068 G0067 G0066 G0064) ((:18) G0065 G0048) ((:
15) G0074 G0073 G0070) ((:5) G0051 G0039 G0038 G0036 G0035) ((:21) G0075 G0051) ((:22) G0052 G0034 G0033 G0031) ((:4) G0
054 G0039 G0038 G0037) ((:36) G0078 G0056) ((:35) G0076 G0056) ((:30) G0079 G0077 G0045 G0020 G0019) ((:29) G0079 G0077)
 ((:28) G0078 G0023) ((:31) G0078 G0023) ((:32) G0080 G0026 G0025 G0022) ((:3) G0081 G0029) ((:33) G0080 G0027 G0026 G00
24) ((:2) G0081 G0030 G0029 G0028) ((:40)) ((:27) G0055 G0025 G0024 G0022) ((:26) G0055 G0027) ((:1) G0030 G0028))
((NIL) (NIL) (NIL) ((:6)) (NIL) (NIL) (NIL) (NIL) (NIL) (NIL) ((:38 :37 :39) G0043 G0057 G0041 G0040 G0057 G0044 G0043 G
0042) (NIL) ((:7 :8) G0049 G0058 G0049) ((:17) G0065 G0059) (NIL) ((:9) G0059 G0048) (NIL) (NIL) (NIL) (NIL) ((:10 :16 :
11 :12) G0068 G0063 G0060 G0069 G0064 G0062 G0061 G0047 G0068 G0067 G0066 G0064) ((:18) G0065 G0048) ((:14 :13 :15) G007
1 G0074 G0072 G0071 G0074 G0073 G0070) (NIL) (NIL) ((:25 :23 :22) G0031 G0053 G0033 G0032 G0052 G0034 G0033 G0031) ((:24
 :5 :21 :4) G0037 G0039 G0036 G0035 G0075 G0051 G0054 G0039 G0038 G0037) (NIL) ((:36 :35) G0056 G0076 G0056) (NIL) ((:20
 :34 :19 :30 :29) G0050 G0021 G0044 G0077 G0045 G0020 G0019 G0079 G0077) (NIL) ((:28 :31) G0023 G0078 G0023) (NIL) (NIL)
 (NIL) (NIL) ((:40)) (NIL) ((:32 :33 :27 :26) G0080 G0026 G0025 G0024 G0022 G0055 G0027) ((:3 :2 :1) G0081 G0029 G0030 G
0028))
LOCAL
(LOCAL ASSUMES (:17) (:9) SAME BODY)
(LOCAL ASSUMES (:9 :17) (:18) SAME BODY)
LOCAL
((NIL) (NIL) ((:6)) (NIL) (NIL) (NIL) ((:38 :37 :39) G0043 G0057 G0041 G0040 G0057 G0044 G0043 G0042) ((:7 :8) G0049 G00
58 G0049) ((:18 :9 :17) G0048 G0048 G0065 G0059) (NIL) (NIL) (NIL) ((:10 :16 :11 :12) G0068 G0063 G0060 G0069 G0064 G006
2 G0061 G0047 G0068 G0067 G0066 G0064) (NIL) ((:14 :13 :15) G0071 G0074 G0072 G0071 G0074 G0073 G0070) (NIL) ((:25 :23 :
22) G0031 G0053 G0033 G0032 G0052 G0034 G0033 G0031) ((:24 :5 :21 :4) G0037 G0039 G0036 G0035 G0075 G0051 G0054 G0039 G0
038 G0037) ((:36 :35) G0056 G0076 G0056) ((:20 :34 :19 :30 :29) G0050 G0021 G0046 G0077 G0045 G0020 G0019 G0079 G0077) (
(:28 :31) G0023 G0078 G0023) (NIL) (NIL) ((:40)) ((:32 :33 :27 :26) G0080 G0026 G0025 G0024 G0022 G0055 G0027) ((:3 :2 :
1) G0081 G0029 G0030 G0028))
LOCAL
(((:3 :2 :1) G0081 G0029 G0030 G0028) ((:32 :33 :27 :26) G0080 G0026 G0025 G0024 G0022 G0055 G0027) ((:28 :31) G0023 G00
78 G0023) ((:20 :34 :19 :30 :29) G0050 G0021 G0046 G0077 G0045 G0020 G0019 G0079 G0077) ((:36 :35) G0056 G0076 G0056) ((
:24 :5 :21 :4) G0037 G0039 G0036 G0035 G0075 G0051 G0054 G0039 G0038 G0037) ((:25 :23 :22) G0031 G0053 G0033 G0032 G0052
 G0034 G0033 G0031) ((:14 :13 :15) G0071 G0074 G0072 G0071 G0074 G0073 G0070) ((:10 :16 :11 :12) G0068 G0063 G0060 G0069
 G0064 G0062 G0061 G0047 G0068 G0067 G0066 G0064) ((:18 :9 :17) G0048 G0048 G0065 G0059) ((:7 :8) G0049 G0058 G0049) ((:
38 :37 :39) G0043 G0057 G0041 G0040 G0057 G0044 G0043 G0042))
LOCAL
SMB
RESULTS
(BODY 1. IS :3 :2 :1)
(BODY 2. IS :32 :33 :27 :26)
(BODY 3. IS :28 :31)
(BODY 4. IS :20 :34 :19 :30 :29)
(BODY 5. IS :36 :35)
(BODY 6. IS :24 :5 :21 :4)
(BODY 7. IS :25 :23 :22)
(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)
NIL
```

RESULTS FOR MOMO

<u>Scene BRIDGE</u> Region :10 gets a strong and a weak link with :4, and that is enough to join them. The same is true for :7.

The links of scene BRIDGE (see 'RESULTS FOR BRIDGE') are discussed and displayed in pages 95-98 , figures 'LINKS-BRIDGE' (page 95), 'NUCLEI-BRIDGE' (page 96), 'NEW-NUCLEI-BRIDGE' (page 97), and 'FINAL-BRIDGE' (page 98).

Because RA and SA are matching T's , two wrong links are placed: one between :22 and :28, and the other between :21 and :29. This is not enough to cause an error, because we need two mistakes (two reinforcing each other), two wrong strong links, to fool the program. But that could happen.

It is interesting to note the way in which the long "horizontal table" :25-24-21-27-9-12 was put together. To this effect, see figures 'LINKS-BRIDGE' and 'NUCLEI-BRIDGE'.

Vertex JB produces only one link between :5 and :8. Vertex KB inhibits the link (through NOSABO) between :8 and :9, and the link between :5 and :9 gets inhibited by S, because it is a T (cf. NOSABO, page 82).

The concave object :7-6-5-4-8-10-11 gets properly identified. We may say that, in general, the more "crooked" or complicated an object is, the easier will be for SEE to isolate it, because there will be many vertices contributing with valuable links.

No mistake was made by SEE on BRIDGE; its eight bodies were correctly identified (see 'RESULTS FOR BRIDGE', page 181).

The background of 'BRIDGE' was also correctly isolated; see that in page 230, section 'On background discrimination by computer'.

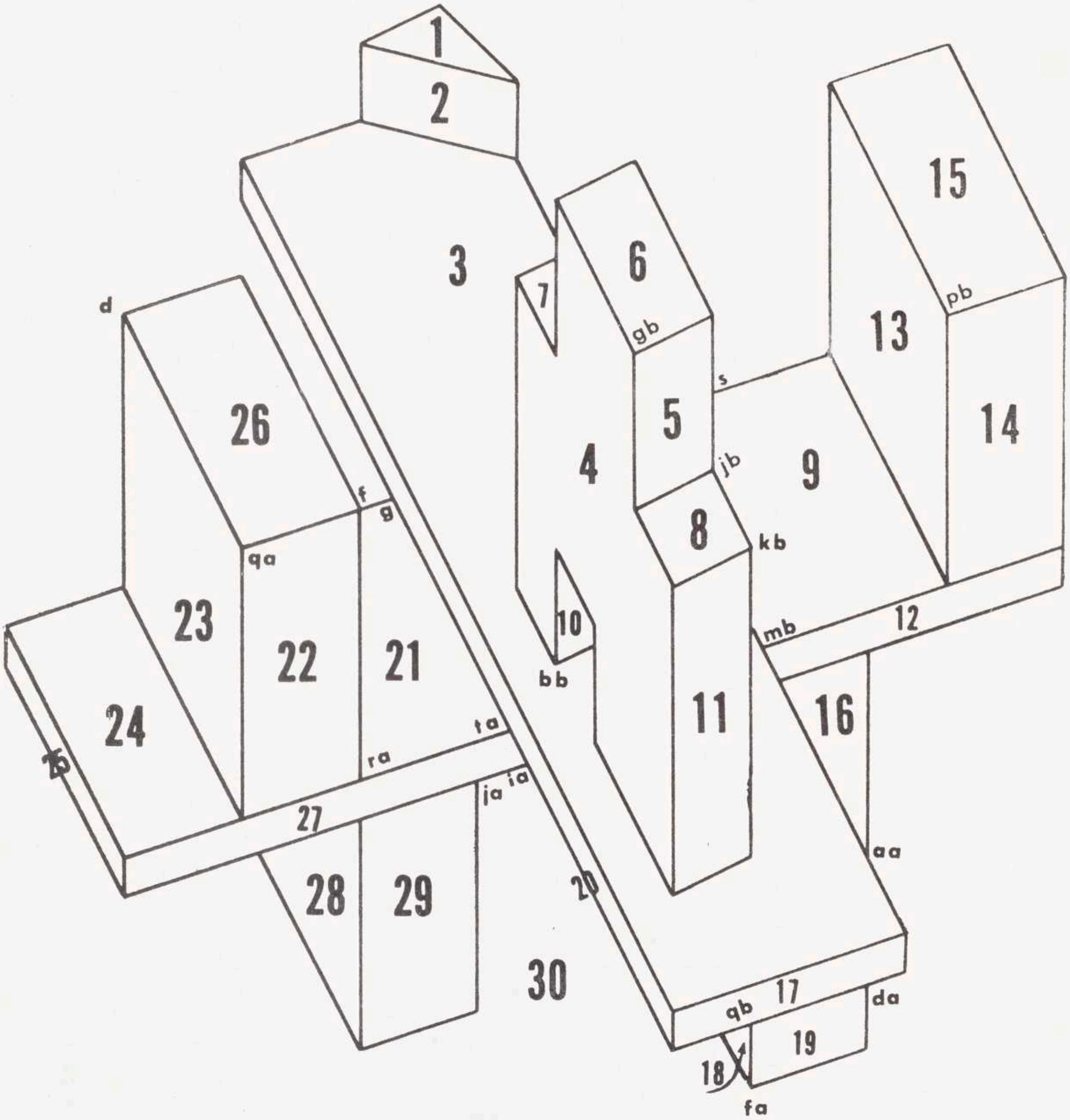


FIGURE 'B R I D G E'

```
SEE 58 ANALYZES BRIDGE
EVIDENCE
LOCALEVIDENCE
TRIANG
GLOBAL
((NIL) ((:8) G0020 G0018 G0017 G0016) ((:7) G0023) ((:10) G0024) ((:11) G0025 G0019 G0018 G0016) ((:28) G0038 G0031) ((:
29) G0038 G0030) ((:27) G0039 G0037 G0036 G0029 G0015) ((:18) G0041) ((:19) G0042 G0041) ((:17) G0043 G0040 G0027 G0026)
 ((:16) G0042) ((:12) G0039 G0015 G0013) ((:14) G0044 G0012 G0011) ((:15) G0045 G0044 G0012 G0010) ((:13) G0045 G0011 G0
010) ((:9) G0051 G0014 G0013) ((:5) G0046 G0022 G0020 G0017) ((:6) G0047 G0046 G0022 G0021) ((:4) G0047 G0025 G0024 G002
3 G0021 G0019) ((:1) G0049 G0048) ((:2) G0049 G0048) ((:3) G0050 G0043 G0028 G0027) ((:20) G0050 G0040 G0028 G0026) ((:2
1) G0051 G0030 G0029 G0014) ((:22) G0052 G0034 G0033 G0031) ((:26) G0053 G0052 G0034 G0032) ((:23) G0053 G0033 G0032) ((
:24) G0054 G0035) ((:30)) ((:25) G0054 G0037 G0036 G0035))
((NIL) (NIL) ((:7) G0023) ((:10) G0024) (NIL) ((:28) G0038 G0031) ((:29) G0038 G0030) (NIL) ((:18) G0041) ((:19) G0042 G
0041) (NIL) ((:16) G0042) (NIL) (NIL) (NIL) ((:14 :15 :13) G0011 G0044 G0012 G0045 G0011 G0010) (NIL) (NIL) (NIL) ((:8 :
11 :5 :6 :4) G0019 G0018 G0016 G0020 G0017 G0046 G0022 G0047 G0025 G0024 G0023 G0021 G0019) (NIL) ((:1 :2) G0048 G0049 G
0048) (NIL) ((:17 :3 :20) G0026 G0043 G0027 G0050 G0040 G0028 G0026) (NIL) (NIL) (NIL) ((:22 :26 :23) G0033 G0031 G0052
G0034 G0053 G0033 G0032) (NIL) ((:30)) ((:24 :9 :21 :27 :12 :25) G0035 G0051 G0030 G0014 G0029 G0039 G0015 G0013 G0054 G
0037 G0036 G0035))
LOCAL
(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)
((NIL) ((:8 :11 :5 :6 :4 :10 :7) G0018 G0016 G0020 G0017 G0046 G0022 G0047 G0025 G0021 G0019 G0024 G0023) (NIL) ((:29 :2
8) G0030 G0038 G0031) (NIL) ((:19 :18) G0042 G0041) (NIL) ((:16) G0042) (NIL) ((:14 :15 :13) G0011 G0044 G0012 G0045 G00
11 G0010) (NIL) (NIL) ((:1 :2) G0048 G0049 G0048) ((:17 :3 :20) G0026 G0043 G0027 G0050 G0040 G0028 G0026) (NIL) ((:22 :
26 :23) G0033 G0031 G0052 G0034 G0053 G0033 G0032) ((:30)) ((:24 :9 :21 :27 :12 :25) G0035 G0051 G0030 G0014 G0029 G0039
 G0015 G0013 G0054 G0037 G0036 G0035))
LOCAL
(SINGLEBODY ASSUMES (:19 :18) (:16) SAME BODY)
(((:24 :9 :21 :27 :12 :25) G0035 G0051 G0030 G0014 G0029 G0039 G0015 G0013 G0054 G0037 G0036 G0035) ((:22 :26 :23) G0033
 G0031 G0052 G0034 G0053 G0033 G0032) ((:17 :3 :20) G0026 G0043 G0027 G0050 G0040 G0028 G0026) ((:1 :2) G0048 G0049 G004
8) ((:14 :15 :13) G0011 G0044 G0012 G0045 G0011 G0010) (NIL) ((:19 :18 :16) G0042 G0041) ((:29 :28) G0030 G0038 G0031) (
(:8 :11 :5 :6 :4 :10 :7) G0018 G0016 G0020 G0017 G0046 G0022 G0047 G0025 G0021 G0019 G0024 G0023))
LOCAL
SMB
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)
(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)
NIL
```

RESULTS FOR BRIDGE

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.

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.

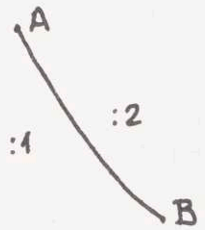
C U R V E D   O B J E C T S

How to extend SEE to work with objects possessing curved surfaces.

Introduction and Summary Most of the heuristics that establish links at each vertex are unconcerned if the edges are curve or straight; a few heuristics get affected: those that use the concepts of collinearity and parallelism.

Thus, it is necessary to redefine and broaden these concepts.

1. A slight generalization is obtained if each segment is represented as having two slopes (initial and final). The functions PARALLEL and COLINEAR of SEE are already modified for this (cf. listings).

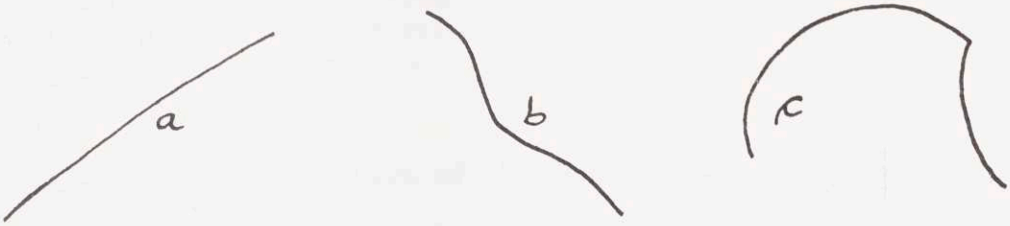


SEE does not care if the line joining two vertices is a straight or curved line. The information about the segment A-B that is relevant to SEE is:
(a) There is a line between vertex A and vertex B.
(b) The coordinates of A and B.
(c) The segment A-B separates region :1 from :2.

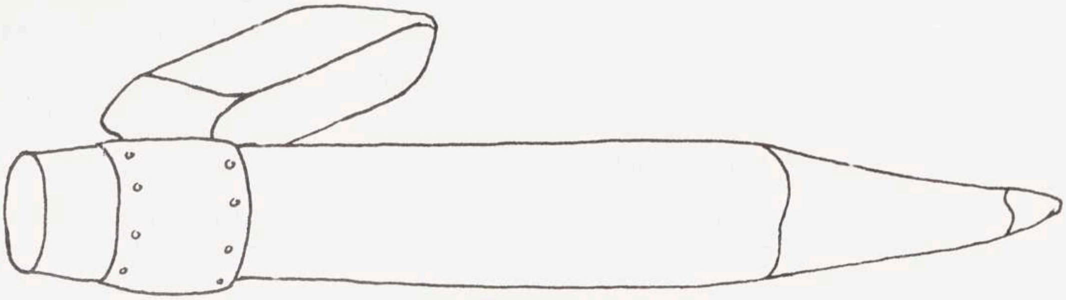
2. Attempts to take limited account of the shape of the segment carry us to

(a) gently bent segments (definition) are those with bounded slope [Bounded curvature will lead to another definition].

A quasi-rectilinear object has faces, vertices and gently bent edges or segments; it is expected that SEE will work well for them. We should try some scenes. **SUGGESTION**

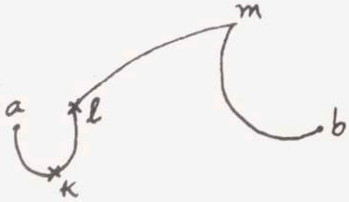


a, b: gently bent segments. c: non-gently bent segment. A gently bent segment has a slope that at any point of the segment does not differ more than epsilon from the mean slope of the segment. All slopes fall in an interval around the mean slope. Gently bent segments form quasi-rectilinear objects.



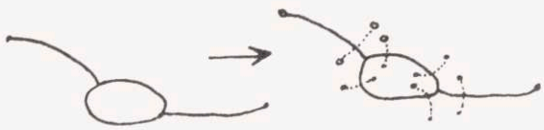
Quasi-rectilinear objects. It is expected that SEE will work well for them.

(b) <u>partition of a non-gently bent segment into several gently bent</u>. Many of the bodies have vertices and curved edges, but the bodies are not quasi-rectilinear (a piece of chewed gum, leaves of a tree). By breaking the edges into gently bent sub-segments, they become quasi-rectilinear bodies. The breaks will occur in points where the curvature is large. There has to be devised a way to break a segment in a unique manner. To avoid breaking a body into two by the introduction of these artificial vertices, we propose to introduce also artificial links between regions, to account for the artificial vertex.



The non-gently bent segment ab gets broken in gently bent segments ak, kl, lm, mb, by the artificial introduction of "new" vertices k, l, m.

Here, the introduction of additional vertices has to be accompanied of 'artificial' or reinforcing links, to preserve the individuality of the body (of the owner of such vertices).

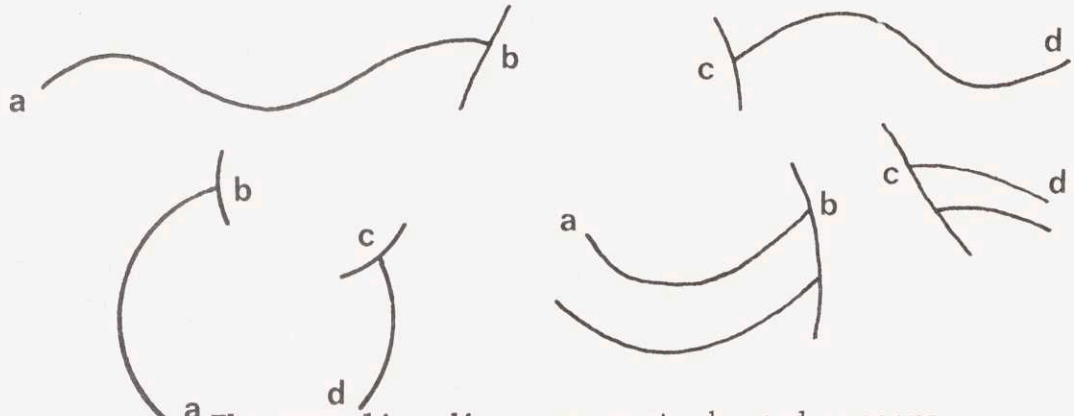


3. More complete consideration of the shape of the segments is obtained as follows:

(a) For parallelism, by requiring that two segments be parallel only if one is a translation of the other. Generally, this is a comparison that takes a time proportional to the length of the segment. Chain encoding {Freeman} {Conrad} is suggested.

(b) For colinearity, by discovering properties or features that "carry through" or are common. Among these are:

1. Mathematical "regularity" of the segments. Both segments are described by the same or similar polinomios, etc.
2. Heuristic properties: there must exist properties which will select with high probability the "right" continuation.
3. Outside of the set of geometric properties, we have color, texture, etc.



The same line dissappears at b and appears at c, making b and c "matching Ts", but to discover this fact it is necessary to have a concept of "good continuation" or "good contour".

Alternatively, we may forget these properties here and include them into <u>models</u> of our curved objects, but then we are forced to make searchs in our scene like those made by DT or TD {my M.S. Thesis}.

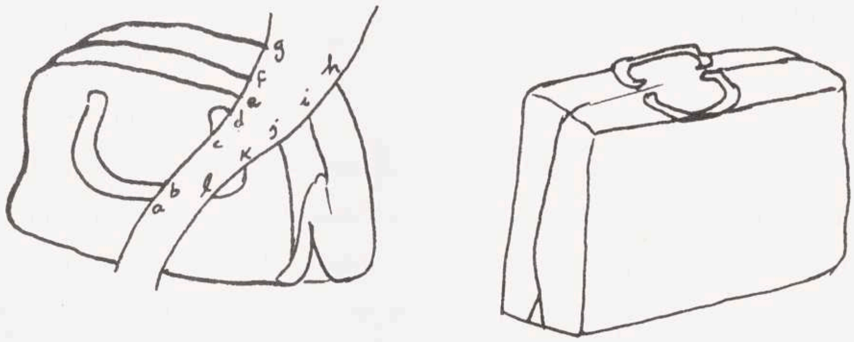


Fig. 'S U I T C A S E S'

Heuristic properties of segments (yet to be determined) could select a "correct" match for endings a, b, ..., k,l.

4. Bodies with no edges and vertices are in principle easily identified. See fig. 'FRUIT'.

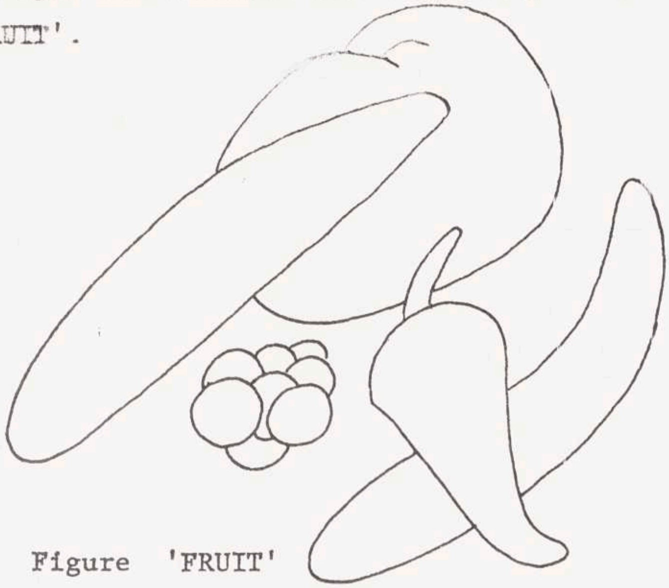


Figure 'FRUIT'

The bodies have no curved edges, and no vertices. The entire surface is smooth; no sharp edges or pointy corners. Examples: an inflated balloon, a frankfurt, a face, a cloud.

It is doubtful that we could do something here with SEE. We could try to postulate "artificial" vertices, using perhaps stereo, at the points where the 3-dim curvature is large, and then postulate lines between such vertices. This looks bad.

Or we could reason as follows: since these objects do not have vertices or edges, then the only vertices appearing in the scene must separate two bodies. They will be mainly T-joints. In principle, separation into bodies looks promising, but recognition (the answer to "what is the name of this object?") seems difficult. Nevertheless, it is not clear that with such a simple set of heuristics we could work successfully with objects as complicated as a human face, a blob of water falling, an amoeba, the surface of the sea (?).

<u>At some point, we have to know what we want</u> As the complexity increases, the concept of "body" depends less and less in geometrical properties (disposition of edges, vertices, ...) and more and more on <u>purpose</u> (Is a skeleton an object? Or perhaps the femur bone alone? The answer varies with our intention -- with the <u>context</u>).

Thus, <u>models</u> are necessary again.

See also 'Do not make over-specialized assumptions...', page 252 .

APPENDIX TO SECTION ON CURVE OBJECTS

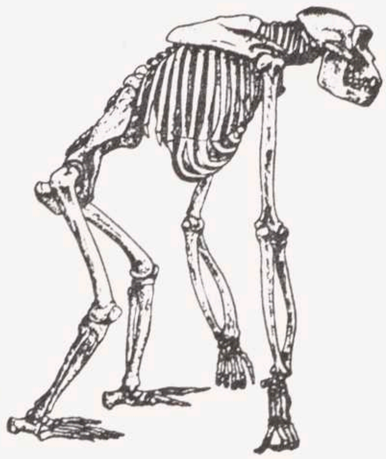
This appendix may be omitted in a first reading.

Requirements for the preprocessor The preprocessor that feeds data to SEE has to find only:

REQUIREMENTS FOR THE PREPROCESSOR

1. The lines of the scene.
2. The vertices.
3. The local slopes at each vertex.
4. See also comments to figure R17.
5. Illegal scenes (page 217) should be detected by the preprocessor.

How bad will be curved objects In objects where the curves edges are gently bent, SEE will work fairly well. The more an edge departs from its rectilinear equivalent, the worse SEE will work; T-joints will be difficult to find, a FORK may transform into a 'T', etc. (I am talking about the current SEE, described in the listings).



Additional information could be used So far, we are trying to identify objects on the basis of form alone, i. e., geometrical considerations. This is asking a machine to do more than a human being does. Ambiguous line drawings, such as ARCH, become inambiguous when we introduce shading, lighting, texture, color, etc. All of these properties could be used by SEE. In fact, consider how easy would be to identify bodies if each one of them is of different color (and we could sense the fact).

Psychological evidence Knowledge of the algorithms used by human beings for shape continuation (page 188) is relevant. We quote from Krech and Crutchfield {1958}:

Grouping by Good Form. Other things being equal, *stimuli that form a good figure will have a tendency to be grouped.* This is a very general formulation intended to embrace a number of more specific variants of the theme, traditionally classified as follows.

1. *Good continuation.* The tendency for elements to go with others in such a way as to permit the continuation of a line, or a curve, or a movement, in the direction that has already been established (see Fig. 37*c*).

2. *Symmetry.* The favoring of that grouping which will lead to symmetrical or balanced wholes as against asymmetrical ones.

3. *Closure.* The grouping of elements in such a way as to make for a more closed or more complete whole figure.

4. *Common fate.* The favoring of the grouping of those elements that move or change in a common direction, as distinguished from those having other directions of movement or change in the field.

It seems plausible to consider that the percepts resulting from all of the above determinants would be such as to meet the criterion of a good figure, that is, one that tends to be more continuous, more symmetrical, more closed, more unified.

Now the reader will see that a difficulty with this general proposition regarding grouping centers on the crucial phrase "good figure." How can we know which

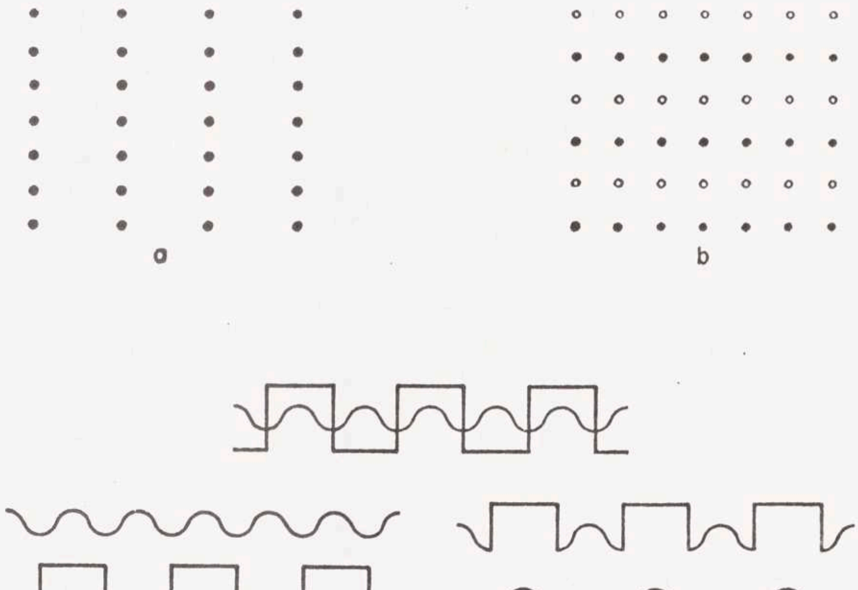
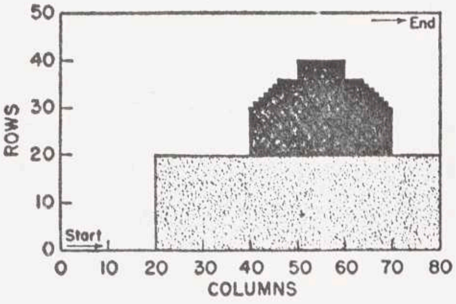


FIG. 37. Examples of grouping. In *a,* the dots are perceived in vertical columns, owing to their greater spatial proximity in the vertical than in the horizontal direction. In *b,* with proximity equal, the rows are perceived as horizontal, owing to grouping by similarity. In *c,* the principle of good continuation results in seeing the upper figure as made up of the two parts shown to the left below, even though logically it might just as well be composed of the two parts shown to the right below, or indeed of any number of other combinations of two or more parts. (Adapted from Wertheimer, 1923.)

BOX 21

How to Measure "Goodness"

Attneave has made an ingenious experimental attack on the problem of measuring the "goodness" of a figure. The subject is given a sheet of graph paper composed of 4,000 tiny squares (50 rows by 80 columns). His task is to guess whether the color of each successive square is black, white, or gray. The experimenter has in mind what the completed figure will look like (fig. *a*).



a

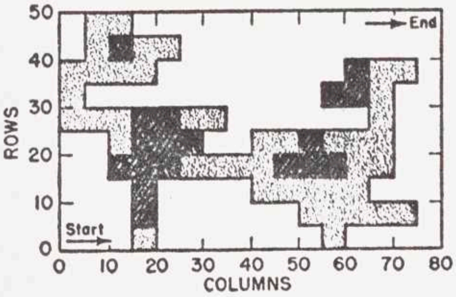
Without knowing what the completed figure will be, the subject starts by guessing the square in the lower left corner. When he has correctly identified the color, he moves on to guess the next square to the right. He continues this process to the end of the row and then starts on the left end of the next row above. In this manner he successively guesses each of the 4,000 squares.

On the average, Attneave's subjects made only 15 to 20 wrong guesses for the entire figure. How was this possible? The answer is that the figure was deliberately designed so that knowledge of *parts* of the figure was sufficient to enable the subject to make fairly valid predictions about the remainder of the figure. This was accomplished by making all the white squares contiguous with one another, and similarly the black and the gray squares. Moreover, the contours separating the white, black, and gray areas are simple and regular. Where the figure tapers, it tapers in a regular way. And it has symmetry; after exploring one side, it is easy to predict the other side. Thus, the subject having discovered that the first few squares are white continues to guess white, and he is correct until he hits the gray contour at the 20th column. After one or two errors, he then continues to guess gray. On the next row above, he tends to repeat the pattern of the first.

All these factors of compactness, symmetry, good continuation, etc., are aspects of what is implied by a "good figure." Thus an objective measure of the "goodness" of a figure is the ease with which the subject can predict its total form from minimal information about a part.

Other figures can be similarly tested. For example, figure *b* would prove to be a less "good" figure because the number of errors in guessing would be larger.

Attneave's particular method will not, of course, apply to all kinds of figures or all kinds of perceptual organizations. But it does demonstrate that there are ways in which "goodness" can be objectively determined.



b

ATTNEAVE, F. 1954. Some informational aspects of visual perception. *Psychol. Rev.*, *61*, 183–93.

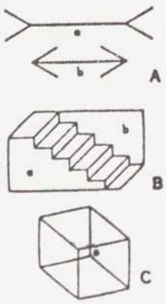
configuration of stimuli is "better" than another?

To escape from this difficulty, we need to have *independent* criteria of what is a good figure. Some approach can be made to this; for instance, in the case of "symmetry" there are objective rules we can apply to determine the relative symmetry of various figures. The same is true of simple cases of "closure." (See Box 21 for a relevant experiment.)

But we are far from being able to state such criteria when we deal with the highly complex configurations of our normal perceptual experience. Part of the difficulty stems from the fact of individual differences among perceivers. One man's mess may be another man's order. And this may reflect the important role of *learning and past experience* in the genesis of "good figure."

ON OPTICAL ILLUSIONS

il·lu·sion \il-'ü-zhən\ *n* [ME, fr. MF, fr. LL *illusion-, illusio*, fr L, action of mocking, fr *illusus*, pp. of *illudere* to mock at, fr. *in-* + *ludere* to play, mock — more at LUDICROUS] **1 a** *obs* : the action of deceiving **b** (1) : the state or fact of being intellectually deceived or misled : MISAPPREHENSION (2) : an instance of such deception **2 a** (1) : a misleading image presented to the vision (2) : something that deceives or misleads intellectually **b** (1) : perception of something objectively existing in such a way as to cause misinterpretation of its actual nature (2) : HALLUCINATION 1 (3) : a pattern capable of reversible perspective **3** : a fine plain transparent bobbinet or tulle usu. made of silk and used for veils, trimmings, and dresses **syn** see DELUSION — **il·lu·sion·al** \-'üzh-nəl, -ən-ᵊl\ *adj* — **il·lu·sion·ary** \il-'ü-zhə-,ner-ē\ *adj*



A

B

C

optical illusions: fig. *A*: *a* equals *b* in length; fig. *B*: either side *a* or side *b* may appear nearer the observer; fig. *C*: *o* may be regarded as either the near or the far corner of the block

Given the nature of SEE, we will restrict the meaning of 'optical illusion' to illusions formed by solids, that is, ambiguities or inconsistencies when we (or the program SEE) try to find 3-dim bodies in a scene; thus, the Müller-Lyer illusion ("A" in the topmost figure) is not considered.

Three kinds of illusions According to this, we may elementarily classify the "scenes that are unlikely to occur" (that is, those that are not "standard" or "normal") in three types:

== Possible but no "good" interpretation.

== Ambiguous -- several good interpretations.

== Impossible: without interpretation.

Like POLYBRICK {Guzmán}, SEE is not especifically designed to handle optical illusions. It was primarily designed to analyze "real world" scenes; hence, an input scene that produces an illusion (in a human) is not likely to occur as input to SEE. Nevertheless, in the same way that we may overtest a program for square roots by asking for the square root of 'APPLE', $\sqrt{🍎}$, we may test SEE with some ambiguous scenes. Let us see what happens.

POSSIBLE BUT NO "GOOD" INTERPRETATION Some objects do not 'make sense' because they violate rules that most objects obey. Nevertheless, it is possible to have "real" three dimensional objects that, when photographed, will give rise to these scenes. Example: Penrose's Triangle.

Over a small region the figure looks normal; the figure makes sense locally. But when we want to incorporate that region in a global picture, conflicts arise. The figure does not make 'global' sense, but each subpart makes 'local' sense, or it can make global sense but not the same sense as the local sense, hence conflicts. Reasoning by the subject does not alleviate totally the situation {Penrose 58}. Mere geometrical considerations do not suffice.

A Dutch artist, Eschen , has greatly exploited this psychological human property.

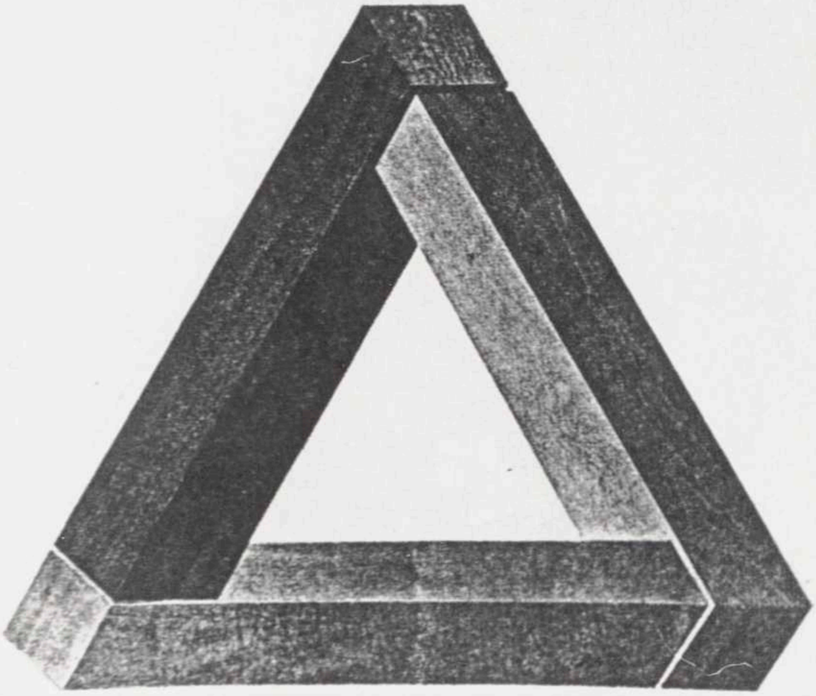
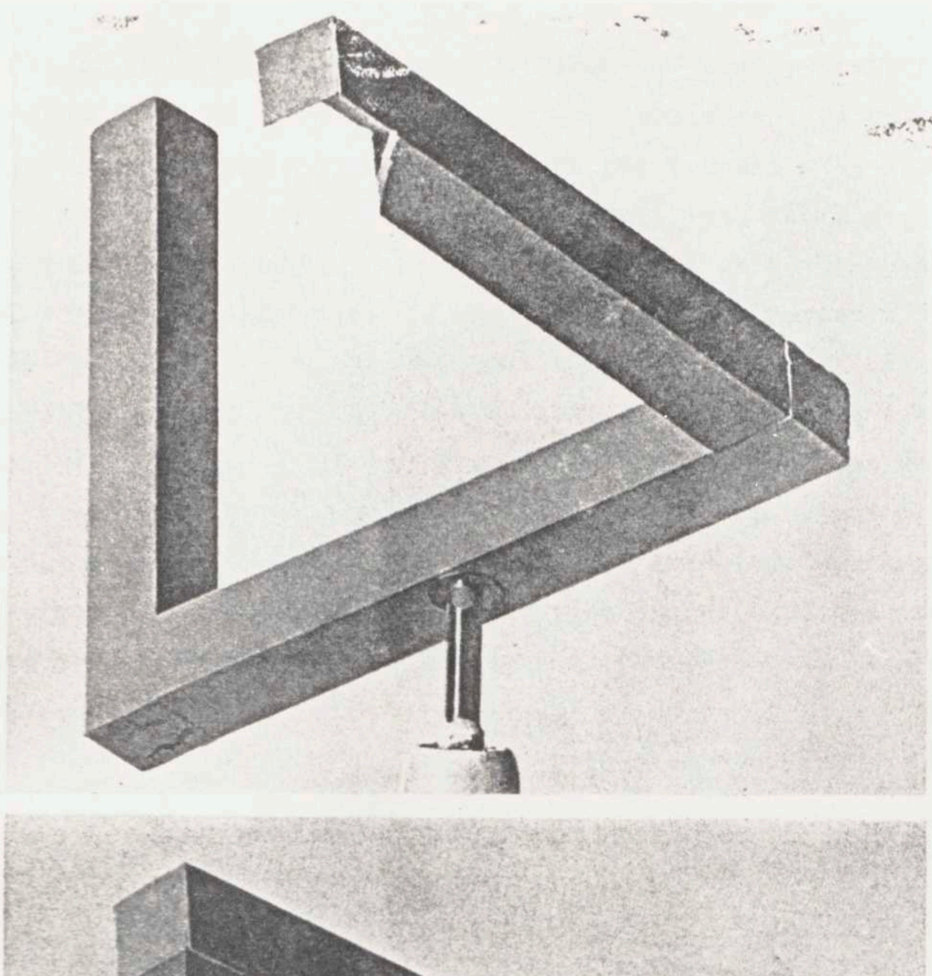


Figure 15. Another 'impossible object'. This triangle cannot exist. (From Penrose, L. S. and Penrose, R. (1958). *Brit. J. Psychol.*, *49*, 31.)
(caption by Gregory {1966}).

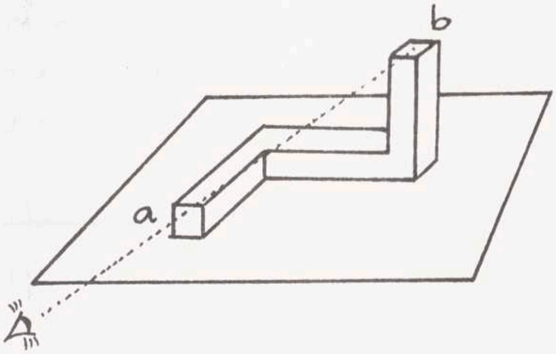
Figure 'PENROSE TRIANGLE'. "IMPOSSIBLE TRIANGLE"

First caption is wrong: there are at least two ways to construct an object that when photographed from certain direction, it will produce this image. One is explained by Gregory in the next page; his object follows quite popular rules and its extrems do not close; other way, this one with a triangle where each side touches the other two (so that the extrems close) is to use curved edges, which nevertheless will look straight from certain direction. See also Metatheorem in page 39 , section 'The concept of a body.'



ACTUAL IMPOSSIBLE TRIANGLE was constructed by the author and his colleagues. The only requirement is that it be viewed with one eye (or photographed) from exactly the right position. The top photograph shows that two arms do not actually meet. When viewed in a certain way (*bottom*), they seem to come together and the illusion is complete. (From Gregory)

One of the strong rules used by humans is that objects whose pictures show straight lines have indeed straight edges; other strong rule is to assume the corners to be like the corners of a cube (faces meeting at right angles)  . Under these rules, the above triangle does not make sense and people will classify it as an "impossible" object ('VARIANT'will be an "impossible" object; Penrose's Triangle will be "3 sticks forming an impossible configuration or scene; "mounted in a funny way"; can not be seen as representing a single object lying in space). For instance, Gregory {Scientific American} tries to explain that the triangle has a real 3-dim object as originator, by constructing a body consisting of three rectangular parallelepipeds ("bricks") joined at right angles, and then taking a picture from a special direction, so that the free ends a and b seem to touch:



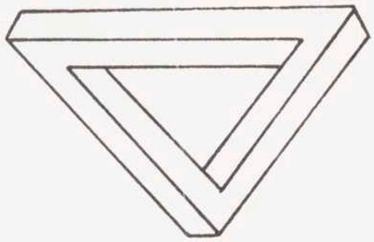


Fig. 'VARIANT'

These rules (faces meet at right angles; straight lines mean straight edges) are deeply ingrained into people, but nature does not need to follow them always. The Penrose Triangle can be obtained by photographing a 3-dim triangle with curved edges and skewed corners, where each side touches the other two.

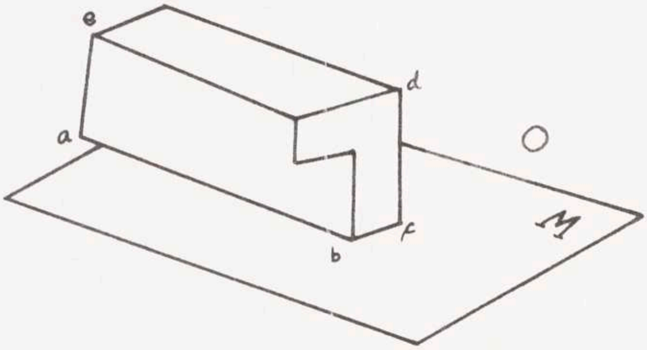
SEE finds three objects in figure 'Penrose Triangle.' Other examples follow.

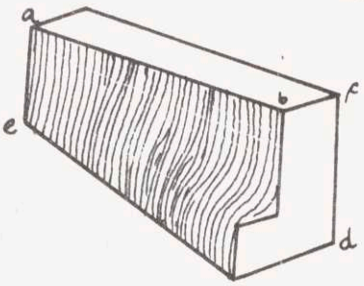
Figure 'B L A C K'

People assume that faces meet at right angles, and this object violates that rule, making it "impossible" or odd-looking.

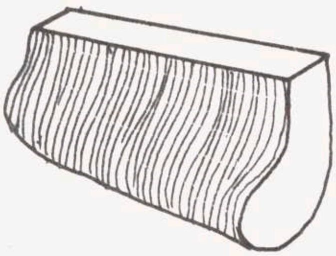
It is possible to construct object 'BLACK' with planar faces. See figure 'TEST OBJECTS' page 209. SEE finds one body in 'BLACK'.

The object at right looks impossible if we assume all faces to be flat. If face aeb is curved, object is plausible. R is its reflection on mirror M, and $\mathcal{R}$ an smoother version of R. $\mathcal{R}$ looks "normal"; by deforming $\mathcal{R}$ we could obtain R.





R


$\mathcal{R}$

Unlike humans, SEE does not hold these "very common rules" as inviolable; SEE does not have any special problems with these "strange but true" objects.

A misleading suggestion of superiority should not be concluded from these rare cases; in other situations SEE makes mistakes that a human being does not (see figure 'SPREAD').

Of course, SEE holds its <u>own</u> rules (for example, those of table 'Global Evidence') as inviolable; hence, given a "rare enough scene" it will make mistakes (cf. assertion in page 51 , after the Theorem). This is a similarity of behavior, I think, between people and SEE -- each one follows rather rigidly a small set of rules. (see also conclusion at end of section).

Besides, often humans will see the 'impossible' object as <u>an object</u>, doing SEE's job just as well.

Figure
'STAIRCASE'

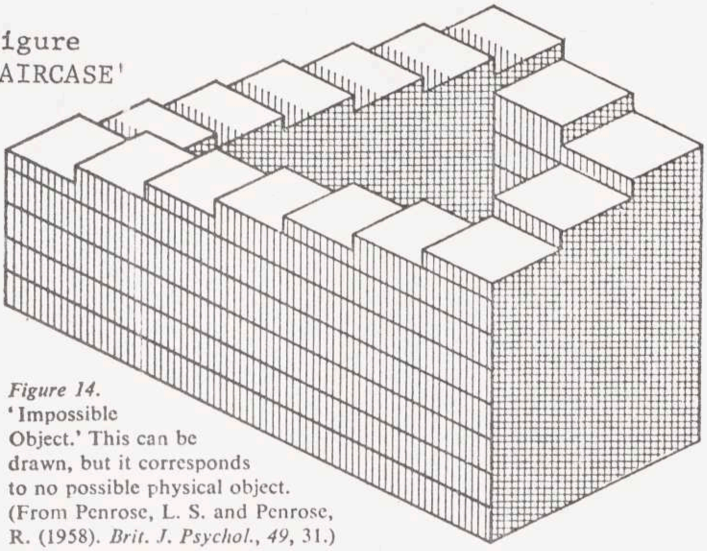
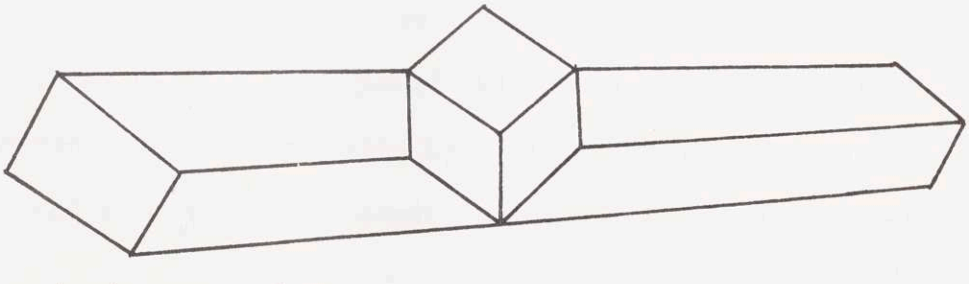


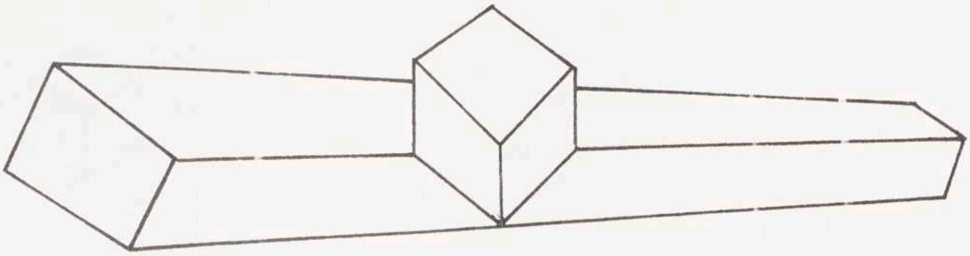
Figure 14.
'Impossible
Object.' This can be
drawn, but it corresponds
to no possible physical object.
(From Penrose, L. S. and Penrose,
R. (1958). *Brit. J. Psychol.*, *49*, 31.)
(caption by Gregory)

The "always descending staircase." {Gregory, in {Foss}}
The caption is wrong, this object could be constructed in the real world, if some surfaces are curved and/or the faces at the corners do not meet at right angles. Example of an object "possible but without 'good' interpretation." See also Metatheorem on page 39 . Again, the "impossibility" or oddness of 'STAIRCASE' comes from assuming the rules 'straight lines in the drawing correspond to straight edges in 3-dim' and 'faces meet at right angles, like corners of a cube' inviolable.

AMBIGUOUS - TWO GOOD INTERPRETATIONS These are scenes that can be interpreted in several correct (non-paradoxical) manners, which are also "sensible" (as opposed to the Trivial Solution of page 41). For instance, an scene like

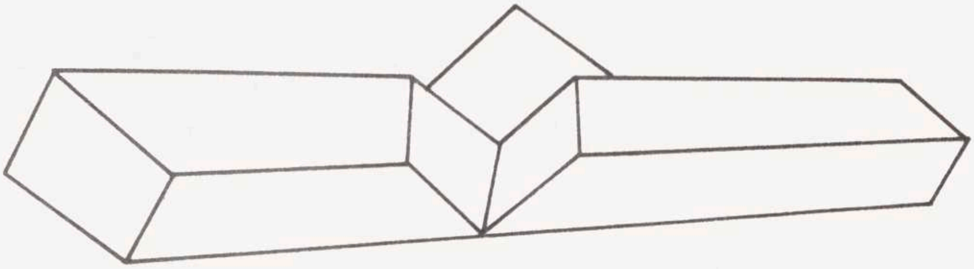


that can be interpreted as



(A)

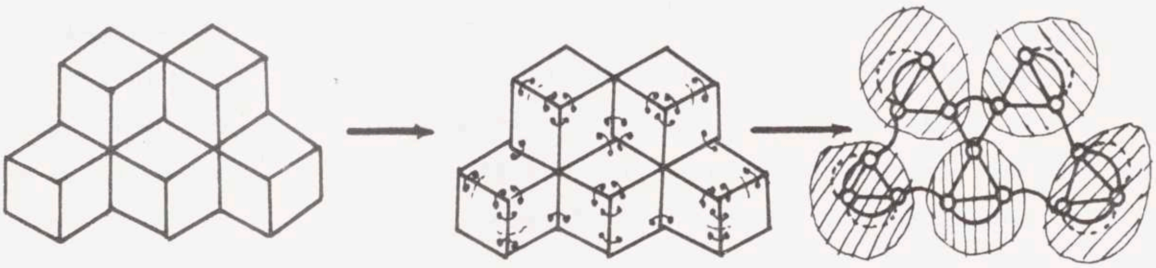
or as



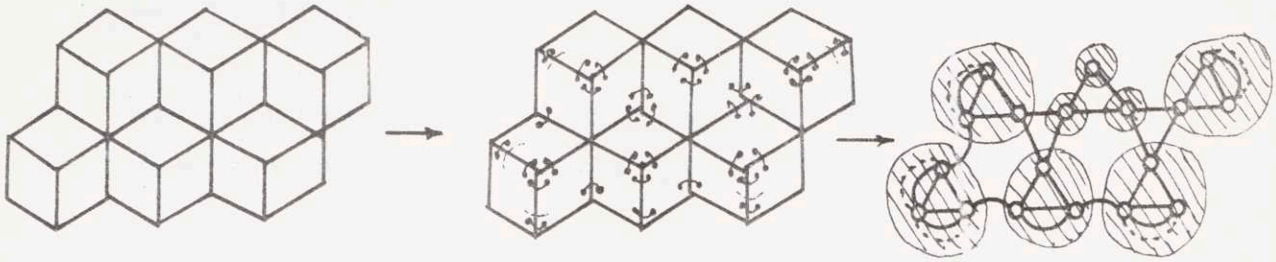
(B)

SEE will generally give one of the possible answers, although not necessarily the one preferred by humans. In this example, SEE chose (B).

The following scene, locally ambiguous, is correctly parsed by our program.

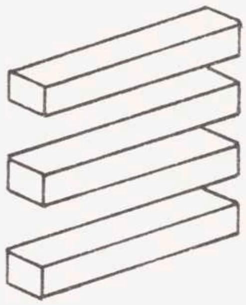


Sometimes, the conservatism of SEE and its partial insufficiency to make very global judgements will leave a body unconnected; for instance, the three faces of one cube below will be reported each one as a separate object, due to insufficient links.



IMPOSSIBLE: WITHOUT INTERPRETATION Images that can not be product of photographing (projecting) a 3-dim scene. These objects do not have physical existence.

This scene is without interpretation, meaning no 3-dim scene (with 3-dim bodies) could have produced it.



In figures like the above one, men are unaware of the extension of the background, and  makes sense even if B is background. SEE is unable to make this mistake, and its analysis of the scene will reflect the fact: the preprocessor will complain that one region, the background, is neighbor of itself. See comments to scene R3, page 113.

Of course, in these cases there is no answer to the question "which are the bodies in the scene?" Whathever answer SEE (or anybody else) gives, it is wrong.

Nevertheless, according to our meta·theorem (page 39), there is an extremely easy way to discover and reject these imposible scenes: all of them are necessarily illegal scenes. And we know how to detect illegal scenes. SEE (or its preprocessor, rather) already does that.

SEE detects all impossible scenes, by refusing the data as an illegal scene.

A PROGRAM TO DISCOVER HUMAN OPTICAL ILLUSIONS

Some scenes get classified by our metatheorem as 'possible but not "good" interpretation', and likewise by SEE, who does not refuse to analyze any legal scene.

Nevertheless, a person will stubbornly classify them as 'odd-looking' or 'not making sense' or 'impossible', even if we teach him the solution obtained by SEE (figures 'Penrose Triangle', 'Black', 'Staircase', 'CONTRADICTORY').

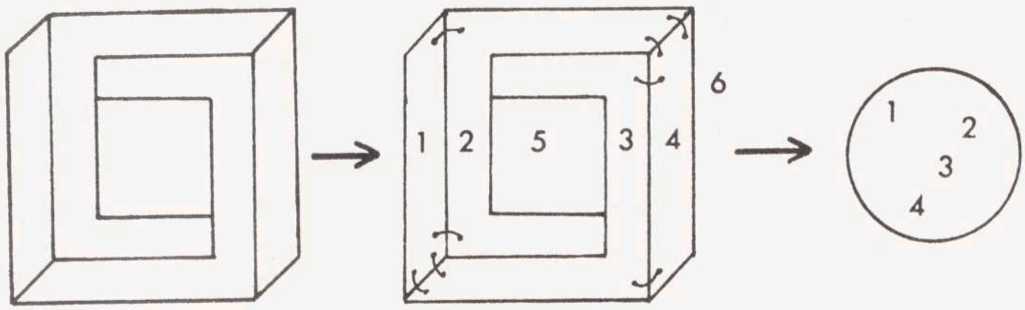


Figure 'CONTRADICTORY'

One object is found by SEE: (:1 :2 :3 :4). As such (since it is a legal scene), SEE classifies it as 'possible but not "good" interpretation'. A person will classify it as "not making 3-dim sense": an human optical illusion. Is it possible to reconcile these views?

Of course, the metatheorem (page 39) insures that there is at least one solution, so SEE's interpretation is "right" (it has chosen one correct answer, generally <u>not</u> the trivial solution given by the metatheorem), and the mortal is wrong. Also, the theorem of page 50 insures that any system (human or computer) that uses too "local" rules (see fig. 'MACHINE') will make at least one mistake, no matter what rules he (or it) uses.

H·optical illusions There is thus a disagreement between SEE and our fellow subject, because SEE has classified the scene as possible but no "good" interpretation' and our man has said 'contradictory as a three-dimensional scene'. Let us call these human optical illusions (such as 'Contradictory', 'Staircase', etc.) by the name h·optical illusions.

What to do in these disagreements? Who is right?

SEE is right Above comments seem to indicate that the electronic data-processor is correct. The human has used excesively "local" rules. That being the case, we can teach and train (if avoiding future errors is desirable) our subjects to "understand", racionalize and make sense out of these h·optical illusions. Indeed, that is what is tried in figures 'Black', 'Penrose Triangle', etc. Different people may show different degrees of (H.optical) illusion before training and after training (see Box). This training is possible (see Box).

In other words, if SEE is right, the computer scientist has nothing to do, it is all up to the psychologists and educators.

Man is right We may hold the view that the human answer is still preferable. Then, to our relief, man is right and SEE is wrong. It is necessary (perhaps) to modify and correct SEE, so as to emulate personal behavior. * We suggest a way to do this.

A program to discover h·optical illusions It is possible to enable SEE to detect these hoptical illusions, so that it will classify the legal scenes into "possible" or "hoptical illusions."

SUGGESTION

As the problem of discriminating between background and objects (see section 'On background discrimination by Computer'), this is an interesting project from the "psychological" point of view but, as in the background case, it is not essential now for our vision-robot work.

* Strictly, there is a third possibility: both are wrong.

BOX

There is generally a wealth of available information—though none entirely reliable—for settling the size and distance of external objects, with sufficient precision for normal use. As is well known, the visual system makes use of a host of 'depth cues', such as gradual loss of detailed texture with increasing distance, haziness due to the atmosphere and nearer objects partly hiding those more distant. These cues were discussed in the nineteenth century by the great von Helmholtz (1925), who fully realised their importance, and they have been the subject of many investigations since, especially by J. J. Gibson (1950). Whatever the richness of depth cues, however, the visual input is always ambiguous. Though the brain makes the best bet on the evidence—it may always be wrong.

The kind of mistakes which occur when the bet is on the favourite though the favourite is not placed, is shown most dramatically by the demonstrations of Adelbert Ames (1946). The most impressive demonstration is given simply with a room which is non-rectangular, but so shaped that it gives the *same retinal image* as a rectangular room to an eye placed in a certain position. Now clearly this room, though queer shaped, *must* appear the same as a normal rectangular room, for it gives the same image to the eye. But consider what happens when objects are placed inside the Ames room. The further wall recedes at one side, so that an object or person standing in one corner is actually at a different distance than is a second object placed at the other far corner. These objects (or people) appear, however, to be at the same distance—and they are seen the *wrong size*. This is clear evidence that we assume rooms to be rectangular (because they usually are) and we interpret the size of objects according to their distance as given by this assumption. When the assumption is wrong we see wrongly. What Ames did was to rig the odds, and then we make the wrong decision on size and distance. A child may appear larger than a man. We may know this is absurd and yet continue to see a bizarre world. The retinal image is all right, but the odds have produced the wrong internal file cards and then the human seeing machine is upset, and gives a wrong answer.

It is interesting that the Ames room is seen correctly by peoples, such as the Zulus, brought up in a 'circular culture' of beehive huts where there are few reliable perspective features, such as rectangular corners and parallel lines, in their visual environment. To the Zulus, the odds are not rigged by the Ames room—to them this is not misleading perspective. They are not subject to this illusion, but accept the room as the shape it is, and see the objects in it correctly in distance and size. This is a matter of very real importance. It shows that when we are transferred to an alien or bizarre environment, where our filing cards are inappropriate, we interpret the images in the eyes according to principles found reliable in the previous, familiar world—but now they may systematically mislead and then perception goes wrong. Space travellers beware! {Gregory, in {Collins and Michie}}

A possible way to attack the problem *is*

(1) To identify each link with whoever proposed it.

(2) To set up systems of simultaneous "symbolic" equations.

(3) To solve them by e limination.

We elaborate:

(1) Mark each link with the name of the heuristic that produces it. After obtaining the 'maximal' nuclei by GLOBAL and LOCAL, seve ral links are left (for example, three in fig. 'FINAL-BRIDGE') and ignored by the current SEE. Instead, one could see what kind of links they are, and one has in this way more information about the type of contradictions in the scene.

(2) Introduce a 'conditional' link: regions :1 and :2 belong to the same body if region :3 does not. An OR link is now possible by use of the conditional, since $a \Rightarrow b \cdot\equiv\cdot b \vee \neg a$.

(2.3) Introduce a 'NOT' link: $:3 \neq :5$, regions :3 and :5 do not belong to the same body.

(2.6) As in ordinary algebraic equations, a system of n simultaneous equations means that all of them must be satisfied; the "AND" of all must be true. Thus, AND is implicit in our notation. So far, we have OR, AND, NOT, IMPLIES (conditional): we have more than necessary.

At the end, we have a system of simultaneous equations like these, where :1 = :2 means both belong to same body; this is an equivalence relation so I use the = sign:

$$:1 = :2 \quad \text{OR} \quad :3 = :5$$
$$:3 \neq :2 \quad \Rightarrow \quad :1 = :4 \qquad (E)$$
$$\ldots\ldots\ldots\ldots$$

We now procede to "solve" these equations. Three things could happen:

== Exactly one solution is found. This is the normal case, and that solution tells what the bodies are. Familiar, "clear", possible scenes will fall in this case.

== More than one solution is found consistent with our equations.

All are reported. This is the case "Ambiguous -- several good interpretations."

== No solution is found. This is a genuine h-optical illusion, corresponding to a contradiction in the equations. For instance, in fig. 'CONTRADICTORY', equations set by the T-joints between :2 and :3 would be inconsistent with those set by the Arrows and Forks.

<u>How to solve equations (E)</u> By the solution to (E) we mean to divide the scene (:1, :2, ..., :n) by means of a partition of the form

(:1 = :5 = :7 = :6),

(:3 = :2),

(:4)

which is consistent with (E).

In the current SEE,

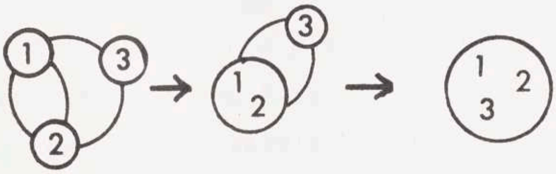
(a) The equations are only equalities: :1 = :2.
Also, equations of the type :1 $\neq$:2 are taken into account by inhibitory mechanisms, such as NOSABO.
No conditional links exist.

(b) Since all equations are of the type :2 = :3, the solution is obtained by applying transitivity, that is,

1 = 2
2 = 3 $\Rightarrow$ (1 = 2 = 3) parentheses indicate nuclei.

Except that we require two antecedents for application of transitivity (two strong links):

1 = 2
2 = 1
1 = 3
2 = 3
$\Rightarrow$ (1 = 2)
1 = 3
2 = 3
$\Rightarrow$ (1 = 2 = 3)



An exhaustive search (which successively tests each possible partition) of the solution to (E) is impractical except in very small scenes, and heuristic methods are needed.

I suggest to start from the equalities such as 1 = 2
2 = 3

and to form nuclei as with the current SEE, except that at each step we check to see if our current nuclei satisfy all of (E). for disjunctive equations such as " 4 = 5 OR 6 ≠ 7 OR 4 = 6 " we try each branch of the OR in turn, rejecting those who conduce to no solution (this may be pretty combinatorial, too).

Perhaps it is possible to use more Logic here -- some sort of theorem proving.

__Conclusions and conjectures__ The similarities between SEE and people (see also 'Human perception vs. computer perception, page 259) stem from the fact that, like SEE, people seem to use only a small number of rules (although not necessarily those used by SEE), which work in almost all cases, but when these rules conduct to an ambiguity or inconsistency ("conflicts"), there is reticence to abandon them, and mistakes or impossibilities are produced.

It is possible that, like SEE, people use primarily local clues, and with less frequency more global information to disambiguate interpretations. I think that, in the presence of objects (in 2-dim line drawings, such as 'MOMO', for instance) not seen before, humans follow general rules not unlike those used by SEE to distinguish or decompose a scene into bodies. Rules that apply to all polyhedra have to be invoked, since in presence of previously unseen objects, humans can not use a __model__ of the object.

The more familiar an object is (or if we have reason to suspect it or expect it), the faster we abandon the general rules and propose its model as a possible explanation of part of an scene; we then jump to a model matching routine (__a la DT__ {MAC TR 37}) that tries to fit the model to part of the scene (to a semi-isolated body); general rules __a la SEE__ prevent us from overflowing with our model into other bodies, and help us to deal with partially occluded bodies.

As a future work, I would like to propose a program that matches models to bodies in scenes using both general rules *a la SEE* and particular information in the model *a la DT*, jumping back and forth among the DT behavior (or mode of operation) to the SEE behavior, either to get a new model, to keep matching the current one, or to apply additional general rules to parts of the scene. The serial application of a SEE-like program first, and then of a DT-like program (but able to handle partial matching for partially occluded bodies), may successfully find the object we describe by its model. Mistakes that are likely to occur in the serial (also called 'horizontal') approach can be eliminated by the suggested "jumping back and forth", or duality of behavior.

SUGGESTION

- Scenes
  - familiar or "clear": 'TRIAL'
  - unfamiliar
    - possible but no "good" interpretation (some h·optical illusions). "odd-looking" scenes: 'VARIANT'
    - Ambiguous. Several good interpretations. (example: at bottom of page).
    - impossible - without interpretation (illegal scenes)
      - some h·optical illusions (bottom of page).
      - no illusion: 

ON NOISY INPUT

The performance of our programs is analyzed when the data has imperfections consisting of (1) misplaced vertices, (2) missing edges, (3) spurious extra lines, (4) missing faces, (5) two vertices merged.

The section 'Analysis of Many Scenes' contains results of SEE when applied to imperfect scenes.

Summary It is easy to predict the operation of SEE when the two-dimensional data supplied is clean, in the sense of being an accurate representation of the three-dimensional scene.

In practice, of course, errors will occur in the data and becomes important to know how sensitive is our program to them.

SEE has some serendipity. Many of the imperfections in the data do not cause mistakes in the linking procedure, or the link misplacements are not enough to cause erroneous identification. But mistakes are made.

Here is how different types of imperfections are handled:

== Assignment of types to vertices is highly insensitive to errors in the position of each vertex, except T's that become Forks or Arrows. Two cures to the exception were found, only the first is implemented:
(1) Allow tolerances in concepts of parallelism and colinearity.
(2) A long but slightly twisted rectilinear segment can be "straightened", as indicated in comments on scene R17.

== Missing edges are subdivided in three classes; two of them produce recoverable or detectable errors (hence, susceptible of correction or prevention). It will be difficult to detect if a segment of the third class is missing; these will produce recognition mistakes.

== Additional lines, like the ones caused by edges of shadows, are not easily detected as spurious or superfluous. Their presence mainly produces a di minution in the number of useful links, thus causing sometimes too conservative behavior. -- i.e., proposition of too many bodies.

== Whole faces may be missing. Ordinarily (see scenes L2, L9T),

the remaining part of the body gets correctly identified.

OBTAINING THE DATA

The scenes analyzed by our program in this thesis were obtained by one of two methods:

By free drawing A line drawing representing three-dimensional objects was made; the coordinates of each vertex were accurate measured (or computed) and the information was put in the 'Input Format' form previously described. Also the regions belonging to the background were indicated as such.

These scenes have mnemonic names such as TRIAL, BRIDGE, etc.

What kind of projection did you use? Were these isometric drawings? Since no assumption is made on the rectilinear objects being drawn, the drawings are not isometric, or perspective, or ... projections. They could be any of them. It is not assumed that "we are dealing with prisms, with faces of a body meeting at right angles (like the corners of a cube)," or with convex objects. Neither the drawings nor the program make any assumption of this type. If the reader wishes to adopt the assumption specified above in quotation marks, then the drawings will correspond to orthogonal projections of three-dimensional scenes.

No support hypothesis is needed: if necessary, the objects could be floating in a transparent fluid of their same density.

By construction Arbitrary but not too complicated objects were cut from pine wood, with flat surfaces, and painted black. Their edges were painted white. By placing them on a black table (see first few pictures of this thesis) in different positions and combinations, three-dimensional scenes were created (see figure 'TEST OBJECTS'). With high contrast film pictures were taken, slightly under-exposed so as to render black everything but the lines. Diffuse illumination eliminated shadows [Great help was received in the pictorial task

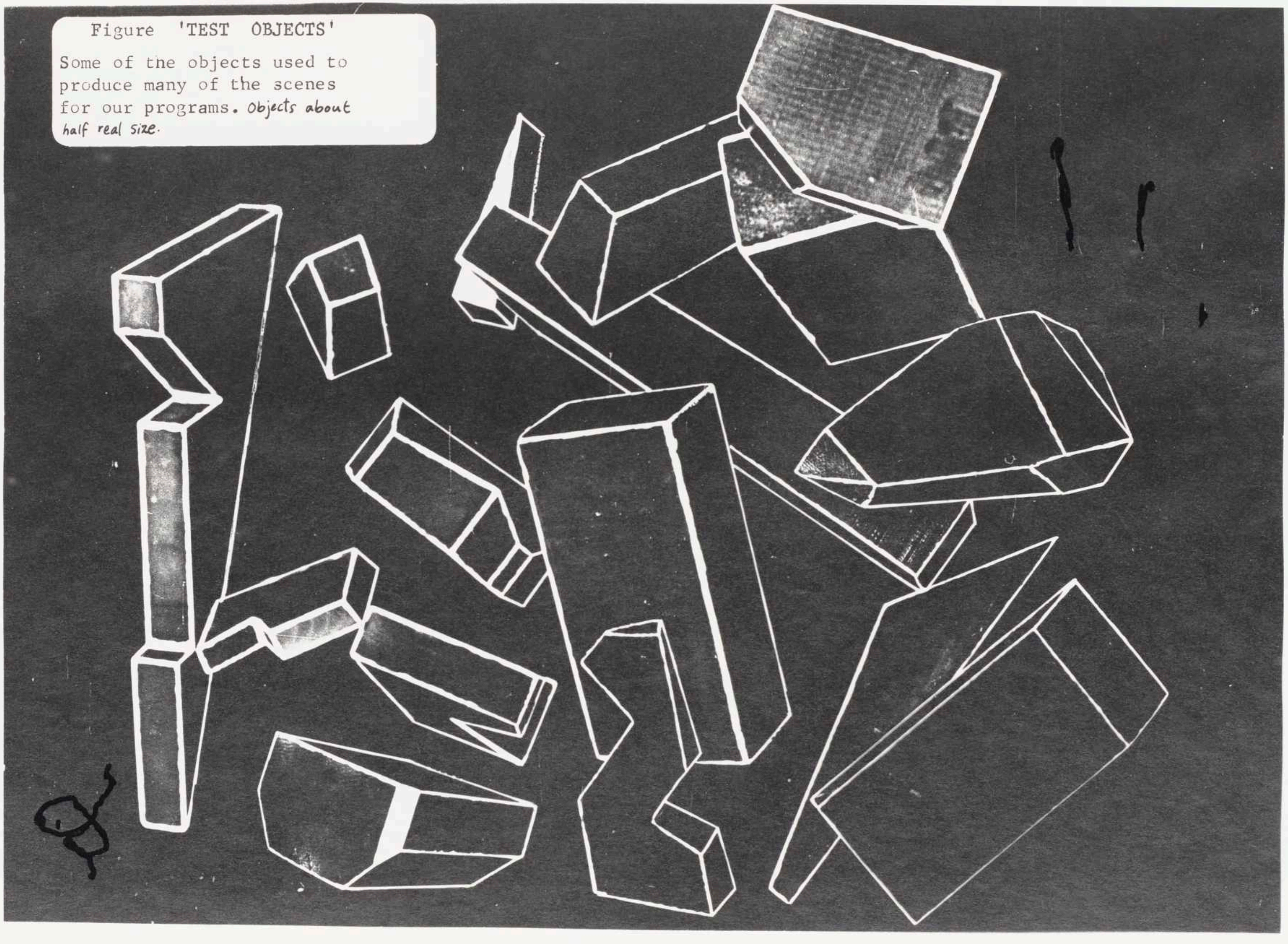


Figure 'TEST OBJECTS'

Some of the objects used to produce many of the scenes for our programs. Objects about half real size.

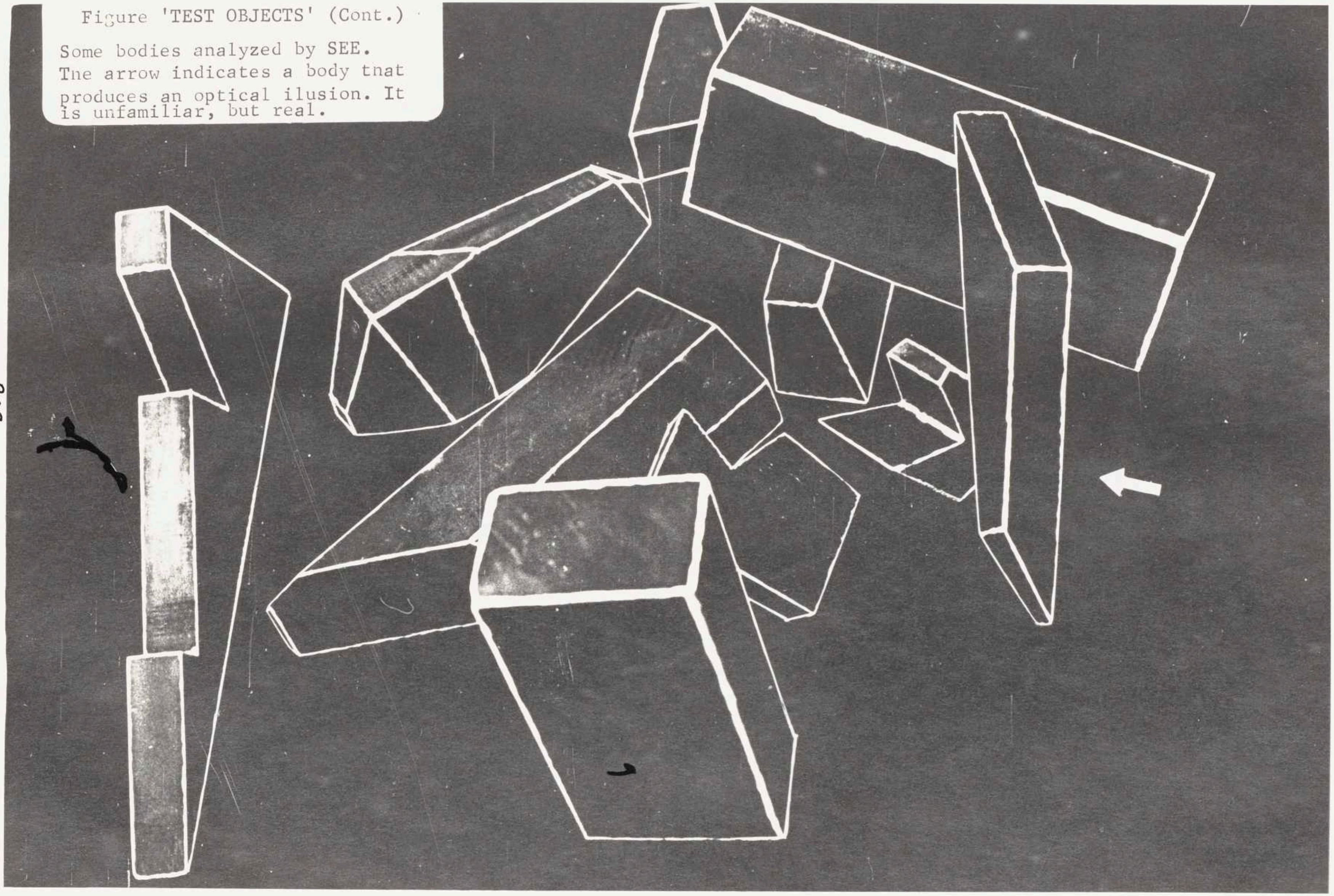


Figure 'TEST OBJECTS' (Cont.)

Some bodies analyzed by SEE. The arrow indicates a body that produces an optical ilusion. It is unfamiliar, but real.

from Messrs. William H. Henneman, Devendra D. Mehta and David Waltz, and is here acknowledged]. The photographs were taken with a depression angle from 45° to 90° (that is, looking down), 50 mm focal length lens, 35 mm camera (standard equipment).

The size of the prints is approx. $8\frac{1}{2}$ by 11 inches (21.5 by 28 cm). If some lines were not clear, they were retouched with white ink. If some lines were missing, they were NOT added.

The pictures have names like L2 or R3, a letter and a digit. Most of them are stereographic pairs, taken with both cameras having parallel optical axes, and the sensitive film on the same plane. SEE only analyzes one scene at the time, so the left picture is not consulted when SEE analyzes the right picture, and viceversa.

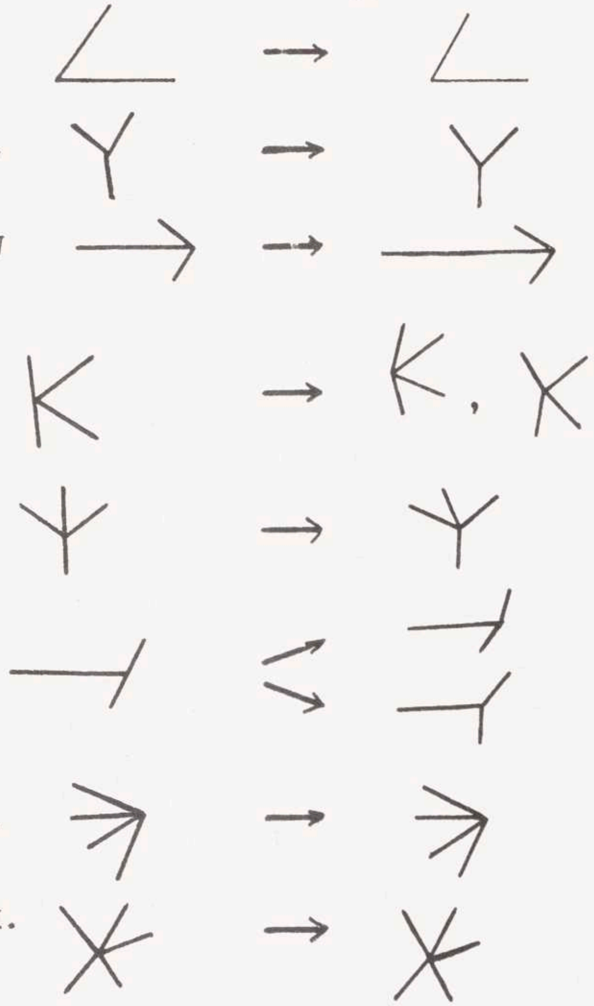
A transparent millimetric mesh is laid on top of the prints, and the coordinates are read by eye and put by hand in the 'Input Format' form. The thickness of each line is about 1 mm (see figure 'TEST OBJECTS'); typically, the size of a scene is 10 or 15 cm: a minimum error of $\pm$ 1 per cent in the coordinates of a vertex is already present. The slopes and directions of short segments suffer, naturally, much greater errors. Also, if two vertices are too close together (about two millimeters) they are merged and codified as one. We are simulating the kind of mistakes that are likely to occur.

Also, some bias is introduced, no doubt, by the human operators. [By reading the coordinates in most of the scenes, immense help was given by Miss Cornelia A. Sullivan and Mr. Devendra D. Mehta; the author acknowledges it.]

Irrespective of the generation method, the scenes that appear in this thesis were drawn in their final form by the PDP-6 computer through a Calcomp plotter, and then inked and finished by hand. Thus, it is possible to perceive in many of them the imperfections of the data that SEE had to analyze.

MISPLACED VERTICES

The coordinates of a vertex may contain a small error or 'noise'. How does this affect the type of a vertex? Does the type change?



Type	Effect
L.	Not affected
FORK.	Not affected
ARROW	Not affected
K.	Transforms into MULTI.
X.	Transforms into MULTI.
T.	Transforms into ARROW Transforms into FORK.
PEAK.	Not affected.
MULTI.	Not affected.

Many types are unaffected. Type K vertices transform into MULTI, but since K's are seldom used by SEE, this is no big loss.

X's transform into MULTIs, and we lose two links here, which makes SEE to behave more conservatively. Also GOODT gets affected (not much).

The serious change are the T's that get transformed into ARROWs or FORKs, when these T's are matching T's. Because they are used for linking otherwise disconnected pieces of a body, their loss generally implies the partition of a body into two. See figure 'DISCONNECTED'.

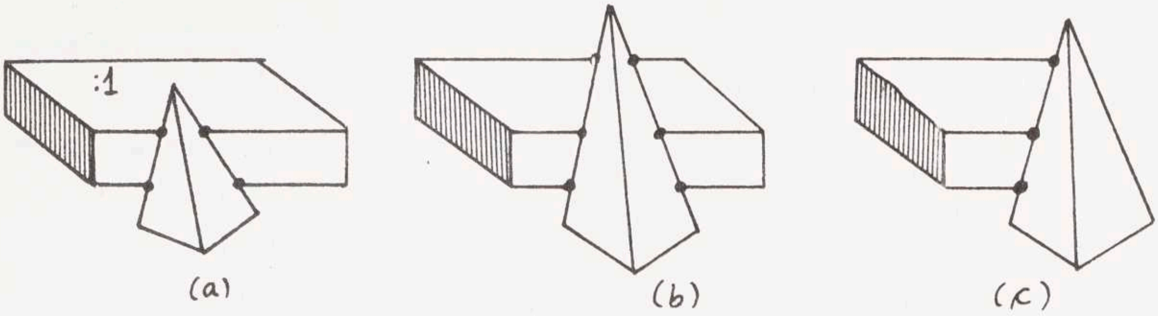


Figure 'DISCONNECTED'

The T's under discussion are marked by small circles (●). In (a), the misclassification of these T's into Arrows or Forks does not break the occluded body, who retains its unity thanks to :1. In (b), the same mis·classification <u>does</u> break the occluded body, reporting two objects instead of one, a possible but less desirable answer. If the T's are not matching T's, as in (c), their mis·classification does not matter.

The loss of matching T's makes the program to be more conservative in some cases. In some sense (see 'Desirability Criterion') this is tolerable.

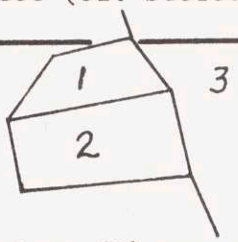
What other perils does the misclassification of the T's bring? We should worry if, due to errors caused by T's, the occluded body joins the occluding one.

DESIRABILITY CRITERION.

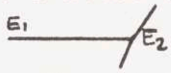
(1) We would like a SEE that never makes mistakes. Since this is not possible, then

(2) We would like it to make mistakes of only one kind, either joins two bodies that should be left separated (intrepid, cavalier behavior), or leaves unattached two nuclei that should be reported as a single object (conservative behavior).

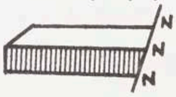
(3) Among the two, we prefer a conservative SEE, because its errors will be easier to correct (cf. Stereo Perception).



The T's should not originate the reporting of :1-2-3 as part of one body

Each T, when perturbed, will go to one of these states: (N) normal, unperturbed; (L) "left", E_2 moves towards E_1, 

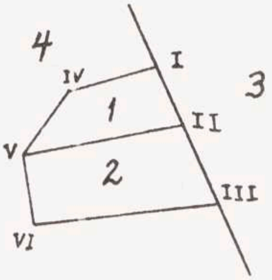
becoming a FORK, or (R) "right", when E_2 moves away



from E_1, 

becoming an Arrow.

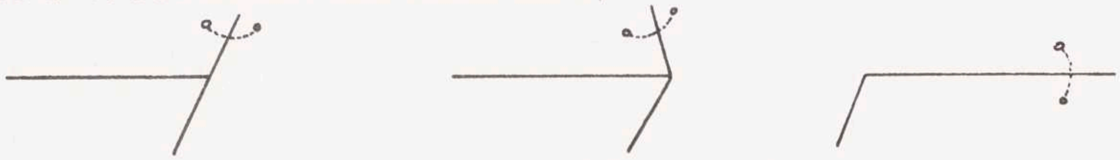
For three T's of an occluded body, $3^3 = 27$ states are possible. They are shown in next page, in table 'THREE Ts'.



How many of these 27 states will produce mis-links joining 1 with 3 or 2 with 3 or 1 with 4 or 2 with 4 (none of the four regions is necessarily background) ?

None.

The reason is that (see description of NOSABO) a T or an Arrow or an L inhibit the link shown below,



so that (a) An arrow in position (I) [or (III)] suggests linking 1 with 4. This link is inhibited by the L at IV [or VI]. Example: Figure R L L in Table 'THREE Ts'. (page 214).

(b) A Fork in position (I) [or (III)] suggests

(i) linking 1 with 3. Inhibited because of the T or arrow in vertex II.

(ii) linking 1 with 4. Inhibited because of the L in IV.

(iii) linking 4 with 3. Depends on outside considerations. Discussed below.

Example: L R L.

(c) An Arrow in position (II) suggests linking 1 with 2. Inhibited or allowed according to vertex V. Example: RRL.

(d) A Fork in position (II) suggests

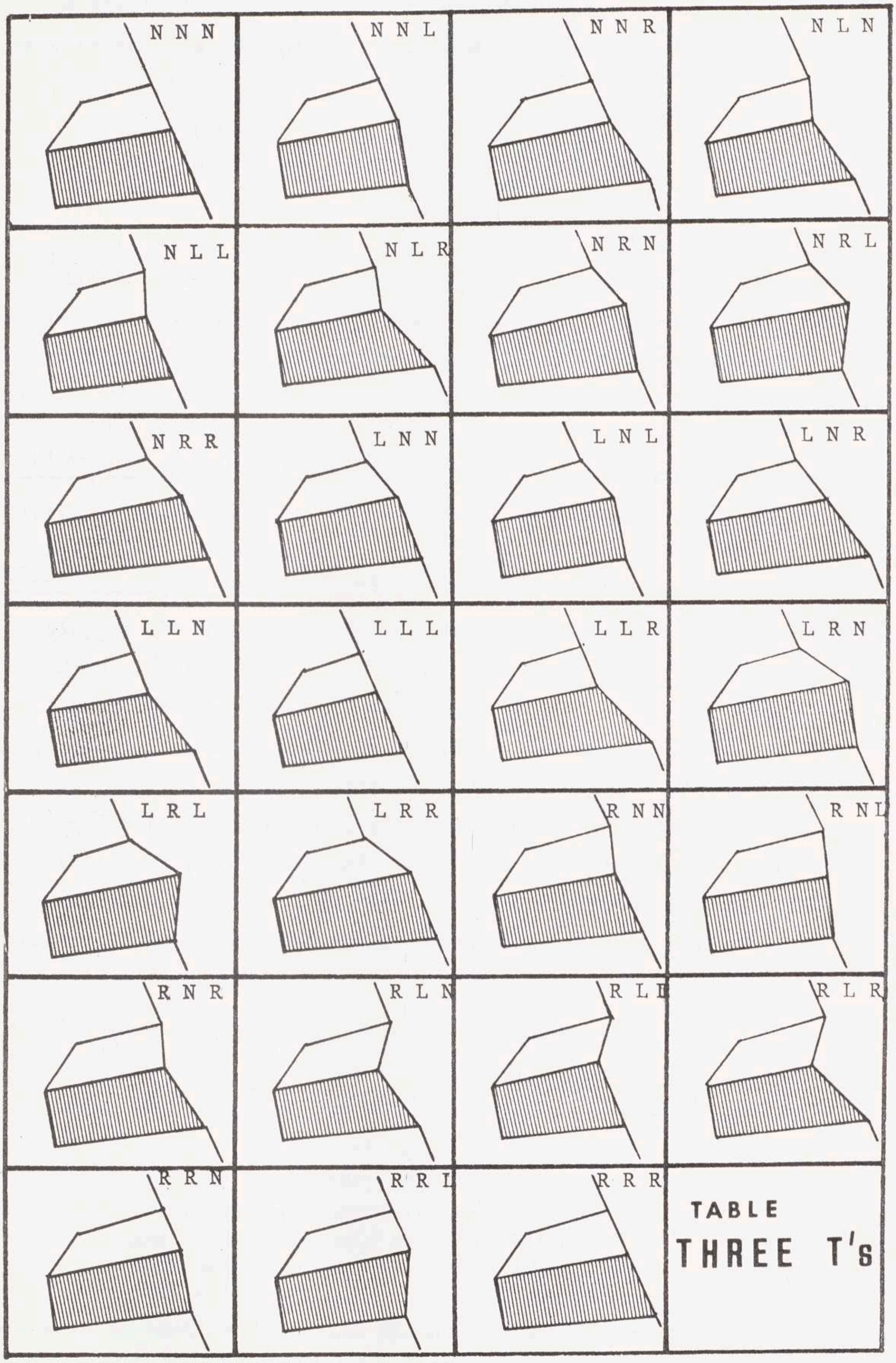
(i) linking 1 with 3. Link inhibited by the T or arrow of I.

(ii) linking 2 with 3. Inhibited by the T or arrow in III.

(iii) linking 1 with 2. Inhibited or allowed according to vertex V.

Example: R L N.

Thus, no link is possible, even under these "noisy" circumstances, between 1 and 3 or 2 and 3 or 1 and 4 or 2 with 4. That is, the 27 cases of table 'THREE Ts' are treated correctly.

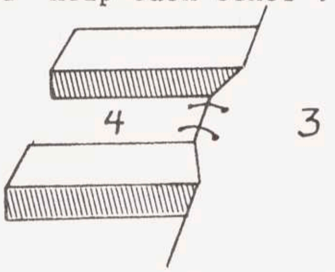


TABLE

THREE T's

A possibility of bad linking exists between 4 and 3 in this case, if two T's convert into forks and "help each other":

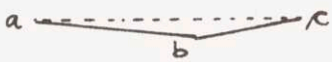
Two links originate the joining of 4 and 3.



Rather than to get involved in this sub-problem, we will point out two solution to the misplaced vertices: (1) by allowing some tolerance in 'parallel' and 'collinear'; (2) by 'straightening out' crooked or twisted segments. We explain.

Equal within epsilon (definition) a is equal within epsilon to b, written $a \stackrel{\epsilon}{=} b$, iff $|a - b| < |\epsilon|$. Generally, $\epsilon > 0$.

Tolerances in collinearity and parallelism Two lines are parallel if the sine of the angle formed by them is smaller than SINTO. ($\text{sine} \stackrel{\text{SINTO}}{=\!=} 0$)
Currently, SINTO = 0.15



Lines ab and bc are colinear if length ab + length bc $\stackrel{\text{COLTO}}{=\!=}$ length ac. Currently, COLTO = 0.05

We have implemented these definitions. Better definitions exist. These definitions allow most small inaccuracies in the coordinates of vertices to pass unnoticed. Although they are giving reasonable service, they are only temporary, since by relaxing too much the criterion for parallelism and collinearity, strange things could happen (fig. 'CROSSED').

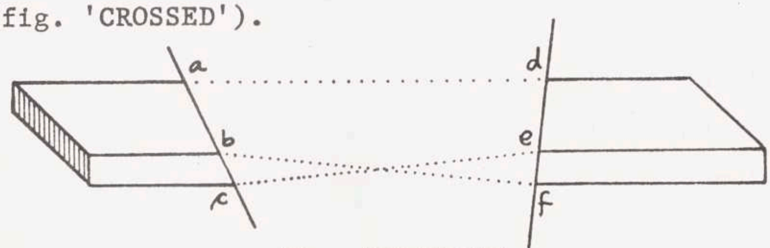
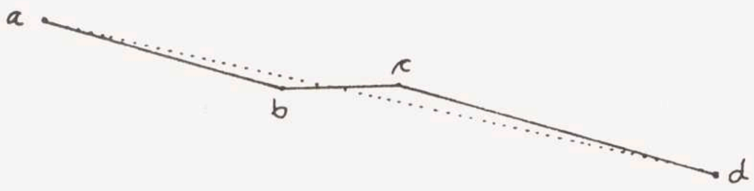


Fig. 'CROSSED'

A too lenient definition of parallel and collinear could give the following matching T's: a to d, b to f, c to e.

See also on section 'Analysis of many scenes' comments to L9 and R9T. (pages 152, 156).

Straightening twisted segments The definitive cure is simple: reassign the slope of bc to be that of ad, if bc is small, ad large



and the angles at b and c are close to 180°. See also comments to figure R17. This has not been implemented. In this way, all cases of table 'THREE Ts' will be solved. See also comments to scene R4.

Probably the preprocessor will automatically take care of this rectification, since it may prefer to give a long segment ad instead of three almost colinear shorter segments ab, bc, cd.

Since the straightening of a segment replaces some known vertices (which we suppose inaccurate) by other idealized vertices, we may be introducing uncertainty, in the form of non verified hypotheses, to our data. The object in the scene could really be "crooked" or twisted.

Fig. 'TWISTED'

The object to the left is really bent as shown. If we idealize it as in the right, we are falsifying the information about it.

By replacing it by an idealized version, we may be creating problems for its identification, when we want to assign a name to it. But notice that the 'unbent' version or idealization is handier for SEE.

If the information is very bad Throw it away and read the scene again. A simile indicates that the issue becomes one of allocation of resources: if you receive a written message containing a few wrong characters and missing words, you may use your brains and time to deduce the omited portions (by employing the redundancy, for instance). If the dispatch is very garbled, you may as well request a new one.

<u>Summary</u> It is known how to handle small inaccuracies in the position of the vertices.

MISSING EDGES

From time to time, and edge will fail to show up in the scene, and the questions are (1) how much harm will be produce, and (2) how can we detect and correct the anomaly. An example appears in page .

<u>Illegal Scenes</u> Lines that end abruptly produce illegal inputs, suggesting that segments are missing.

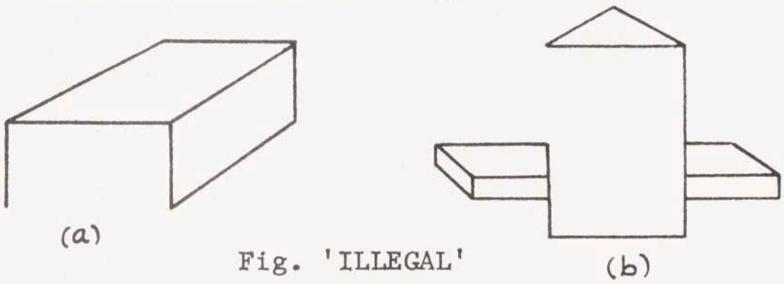


Fig. 'ILLEGAL'

In (a), a vertex has one edge.
In (b), the network can be separated by erasing just one edge.
Both are illegal scenes, indicating missing or extra lines.

Also (Figure 'ILLEGAL', (b)) a region can not be a neighbor of itself -- another irregularity that points to defficient data. Cf. comments to scene R3. (page 113).

These constraints can be nicely exploited by a preprocessor.

<u>Line proposer and line verifier</u> A line proposer is a program that suggests places where a line can be missing; a line verifier is essentially a precise (and slow?) line finder that searches a line in only a small portion of the scene, as told by the line proposer.

In the body of this section we will develop several heuristics for use in a line proposer. The verifier is not discussed.

Blum's line proposer An algorithm has been designed by Manuel Blum {1968}, that will detect many places where lines are possibly missing. It suspects concave regions. An angle bigger than 180^{o} originates a search for the omited line in directions parallel to the neighbor

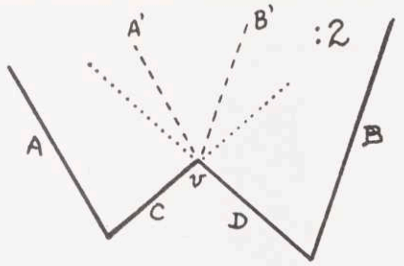


Figure 'B L U M'

> Region :2 is suspected to contain undetected lines, because it is concave. Vertex v is chosen because its internal angle is bigger than 180 degrees. From it, Blum's proposer will suggest to the line verifier to look for lines in directions VA' and VB' (broken lines), parallel to the neighbor edges A and B. It also searches (dotted lines) along the continuation to lines C and D.

edges (fig. 'BLUM'). It also originates searches along its own edges. In other conditions, a vertical line is searched.

No harm is done by a bad proposer. Only some time is wasted.

Internal edges If a missing line is totally internal to a body, and is not detected by the line proposer, its absence will at most cause conservative behavior in SEE. In some cases their absence does not confuse SEE (figure 'MISSING').

The majority of internal edges cause concave regions to appear (fig. 'BLUM'). They will be detected by a line proposer.

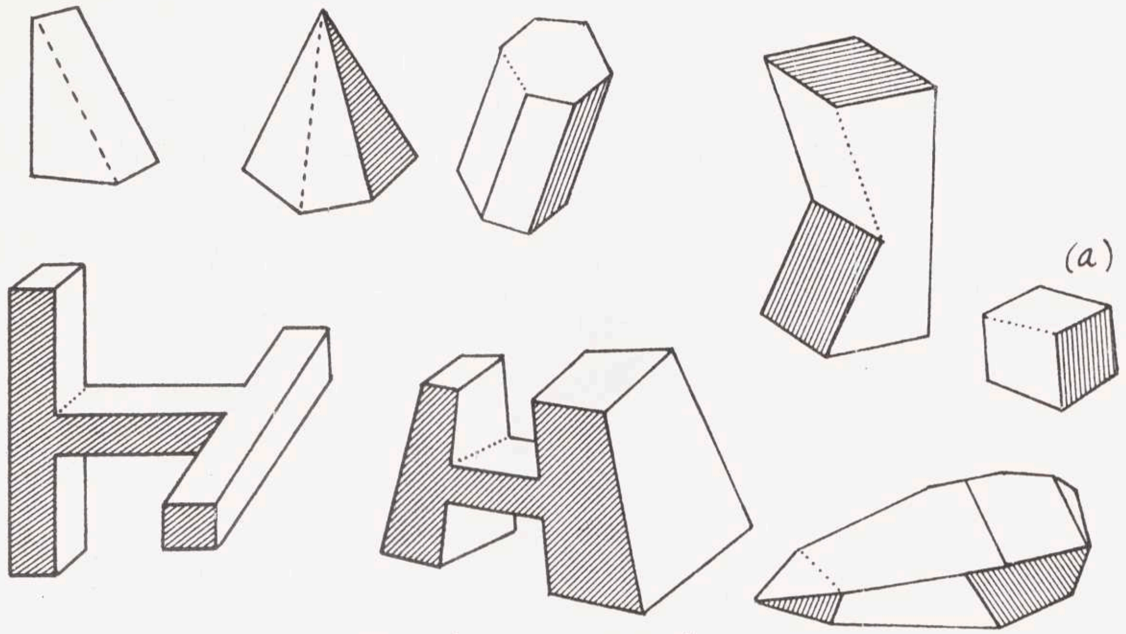


Fig. 'M I S S I N G'

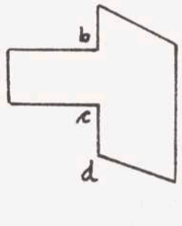
Cases where the disappearance of an internal line (dotted) does not separate the body.
In (a), the object separates into two. This case is recognized by Blum's heuristics. Else, SEE could check for this configuration as a special case.

External edges Edges that separate two bodies are called external. If undetected, their disappearance will cause 'intrepid' errors by SEE, which are undesirable (see 'Desirability criterion'). Two cases result: (1) Only part of the edge disappears; there is possibility of correction. (2) The whole edge is both external and missing (and the scene is still 'legal'): a mistake will occur. See figure 'External Edges'.

Case (1) Only part of an external edge disappears. It can be detected because

(a) a concave region is generated, and

(b) the region has internal angles bigger than 180^o where a line "goes through": ab is colinear with cd.



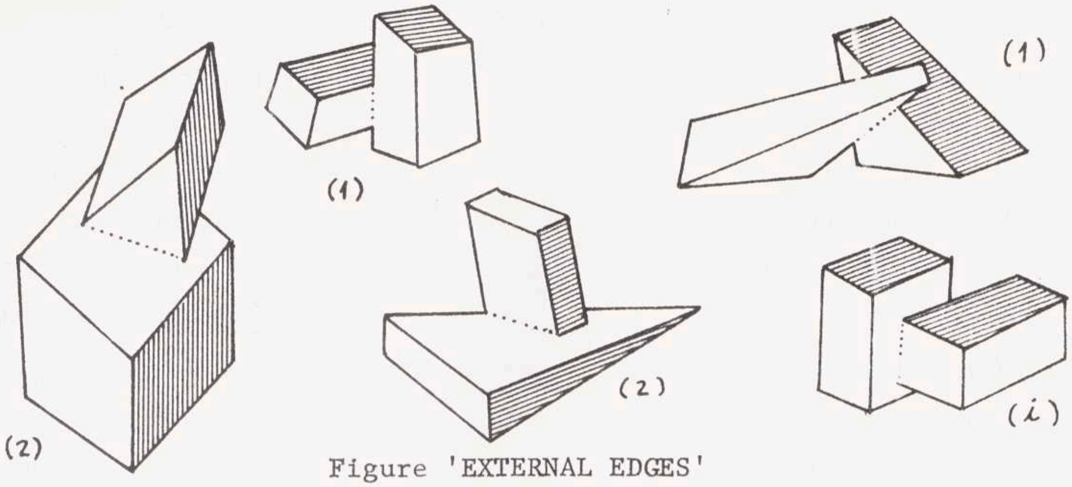


Figure 'EXTERNAL EDGES'

A segment separating two bodies may disappear. (1) If that segment is part of a larger segment, it is possible to sense and correct the anomaly. (2) If a whole external edge is missing, its absence remains undetected, inducing a mistake in SEE. In (i) an external edge disappears, and creates an illegal figure.

Case (2) The complete edge is missing. Then (b) of case 1 fails, and detection is difficult.

SPURIOUS EXTRA LINES

By them are meant lines that "should not be there", such as those caused by edges of shadows.

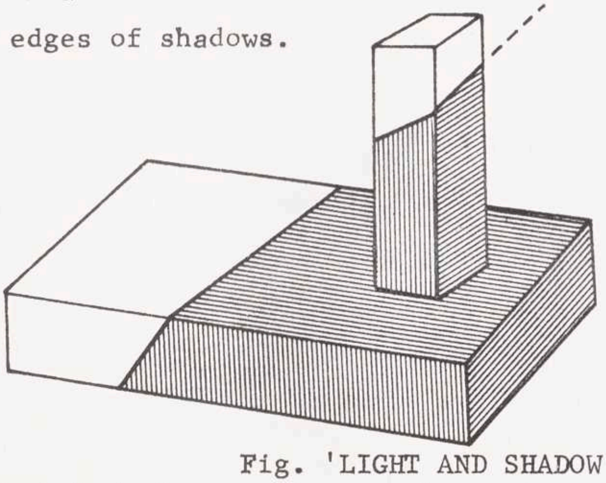


Fig. 'LIGHT AND SHADOW'
Each body becomes two; each one is recognized independently by SEE. Four bodies are found.

Shadows of rectilinear objects travel in planes that (in theory) part an object in two: the illuminated part, and the dark one. Each is a separate object by itself, according to our definition (see 'Several definitions of a body'), since they have plane boundaries. SEE should recognize them.

In practice, we have not tried our program with scenes having lines produced by shadows. A conservative behavior, like in figure 'LIGHT AND SHADOW', is expected.

Some shadows gradually diffuse; multiple lights cause multiple shadows. These problems may have to be solved by assuming or computing the direction or position of the light sources.

MERGED VERTICES

Two vertices fused in one will produce diminution in the number of useful links they report, since the resulting vertex will be of type MULTI. Thus, conservative behavior is expected from SEE in these cases (see Fig. L19, L17T, R17, L4, etc. The program does well in them, when not too many coincidences are present).

It is possible to analyze the vertices of type MULTI and try to decompose them in simpler types (compare figure R19 with WRIST*). Read comments to R19 and L19. **SUGGESTION**

CONCLUSION

On scenes obtained from "real world" data, inaccuracies are expected, and it is required of SEE to work well despite them. Currently, the behavior of the program in these cases is not discouraging, but is not extremely satisfactory, either. The additional work needed depends heavily on obtaining genuine test data, instead of the faked data used in the experiments described.

BACKGROUND DISCRIMINATION BY COMPUTER

A program determines the regions that belong to the background of a given scene; that is, the regions that are not members of any of the bodies. Examples are given.

Need The program SEE requires to know which regions of the scene belong to the background (cf. 'SEE, a program that finds bodies in a scene'). At present, this information is supplied by the user, as described in sections 'Internal format' (page 66) and 'Input Format' (page 63) of a scene.

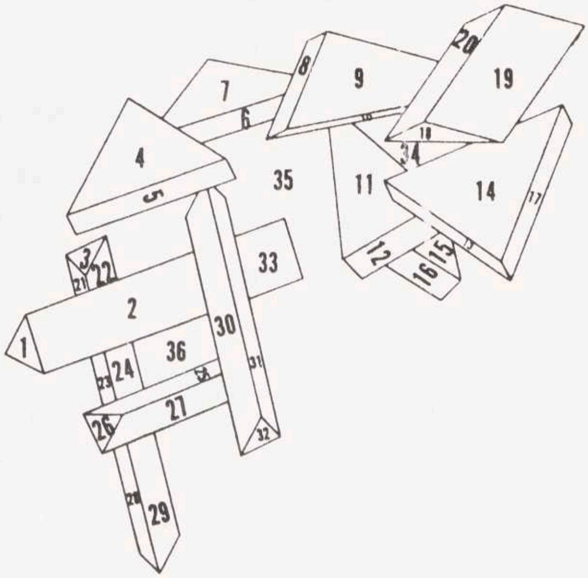
In the current vision experiments, it is not difficult to determine the regions that form the background, since they are always black and homogeneous (see first few pictures in this thesis). But in more realistic scenes, there will be a great demand for a background finding program.

> Therefore, it is interesting to try to develop a program to separate the "ground" in the back from the objects in the "foreground", having a limited information consisting of the scene as described in section 'Internal Format', namely, vertices and edges.
>
> That is, we will use in this task only "geometric" properties.

Such program has been written, and works automatically under the command of PREPARA, the function that converts a scene from its 'Input Format' to its 'Internal Format'. When the regions forming the background are not supplied, PREPARA activates our program, named BACKGROUND, and these regions are searched for; otherwise, SEE is supplied with the background regions as declared in 'Input Format'.

Example. Scene 'HARD'. The results obtained are

```
(SUSPICIOUS ARE NIL)
     THE BACKGROUND OF HARD IS
(:34 :36 :35)
(:34 :36 :35)
```



Three regions are found to be part of the background: :34, :36, and :35. That is correct.

We now proceed to describe the subroutines that make such identification possible.

<u>Suspicious</u> In a first pass, we collect the regions that "may be" background, and call them "suspicious regions". Regions that are not suspicious are LIMPIO (clean).

Ideally, if a region :R contains L's, FORKs, ARROWs or T's in the position below, it is not a part of the background.

(I) (II) (III) (IV)

FIGURE 'BACKGROUND'

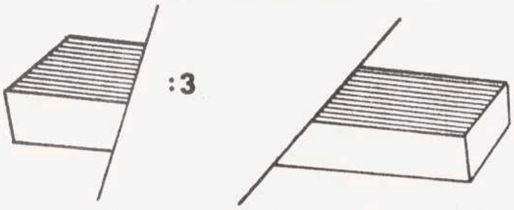
In an idealized situation, :R can not be part of the background: it is <u>clean</u>, or free of suspiciousness. :R will be called 'LIMPIO' (clean).

(I) means that the background [almost] never is the internal part of an 'L' (the region containing the angle smaller than 180 degrees).

(II) means that the background does not contain FORKs.

(III) means that the background is not in the "inside" of an ARROW (the background is not a 'proper'arrow').

(IV) means that the background can not be the flat region of a 'T'; this in turn means that a body can not disappear under the background and then reappear at some other point:



:3 is not the background.

We reinterprete rules (I)-(IV) as follows:

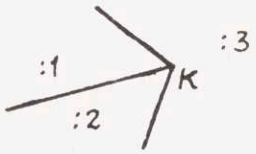
(I) A region "inside" an L is LIMPIO (clean).

(II) A region containing a fork is LIMPIO.

(III) A region "inside" an arrow is LIMPIO.

(IV) A region "on the flat side" of a T is LIMPIO.

Clean Vertex (definition). A vertex is clean with respect to a region if it indicates, through rules I-IV, that such region is LIMPIO. For instance, K is clean for :1 and for :2, since (III) indicates that :1 and :2 are LIMPIO. K is not clean for :3.



These heuristics are not 100 per cent infallible; also, in a moderately complicated scene, coincidences of vertices are bound to occur, originating violations to I-IV. For instance, in figure CORN, vertex UU is a Fork belonging to the background, in contradiction with (II).

For completeness, we present a violation to each one of rules I-IV:

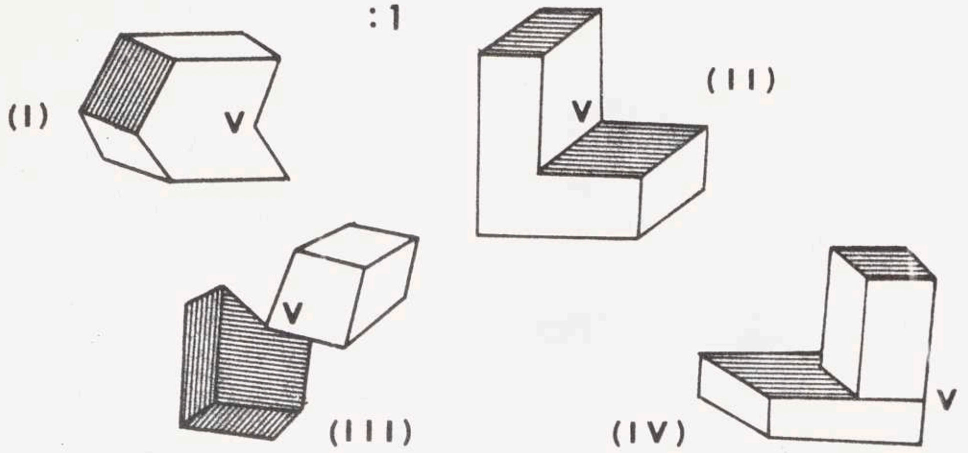


FIGURE 'VIOLATIONS'

:1 is the background. In all four cases, vertex V violates rule specified at the bottom of figure. They are rare cases. The situation indicates that rules I-IV provide noisy information, which has to be dealt with carefully. That is what it is done

The vertices of each region are analyzed under rules (I)-(IV). To allow for coincidences of vertices and rare cases (like those in figure 'VIOLATIONS'), it is permitted for a suspicious region to have a small number of clean vertices.

The number of clean vertices is compared with a quantity that is a small fraction of L (the number of vertices on the boundary); currently, that fraction is L/9.

== If the number of clean vertices, that is, vertices satisfying I-IV is bigger than L/9, we call that region LIMPIO ("clean"). In addition, (a) If L is large (bigger than 25, currently), that region is BIGFACE, such as :21 of scene L19;
(b) Otherwise, is only LIMPIO (normal case).

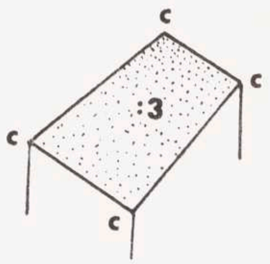
== If it is not bigger than L/9, then it is SUSPICIOUS. Also,
(a) If L is large (bigger than 25), the region is BACKGROUND,
(b) Otherwise is only SUSPICIOUS (normal case).

That is, a region LIMPIO has to have at least

1 + [one vertex of each nine]

"clean" vertices.

Example. Region :3 has four 'clean' vertices (four vertices indicate that :3 is LIMPIO) --- It can not be SUSPICIOUS.



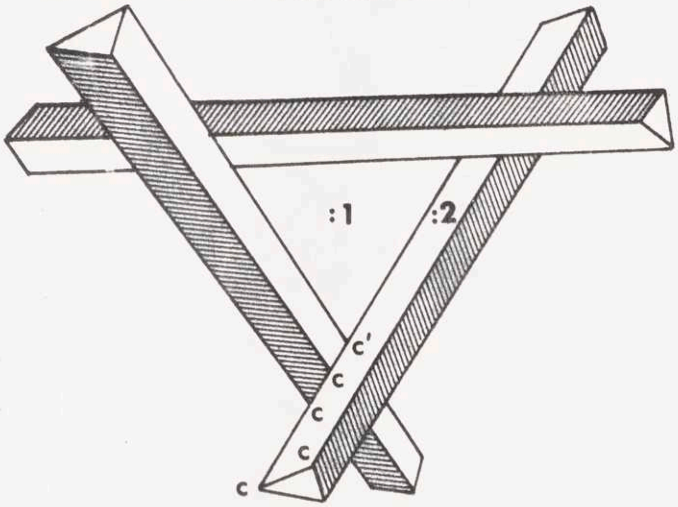


Figure 'EQUILIBRIUM'

(This scene is correctly analyzed by SEE)
All the three vertices of :1 are not clean; :1 will become Suspicious (a candidate for background). Five of the seven vertices of :2 are clean, so :2 is LIMPIO. Note that vertex C' is clean for :2 and not clean for :1.

For example, when we apply the function SUSPICIOUS (see listings) to every region of scene SPREAD, the suspicious regions turn out to be:

Suspicious only: :35 :18 :34 :2 :3 :12 :11 :33 :37 :47 :48 :46.

Background: :48.

<u>Summary</u> By analysis of its vertices, each region is either LIMPIO or SUSPICIOUS. The suspicious regions with more than 25 vertices are classified right away as BACKGROUND: a suspicious region with many edges is probably background.

The selection is done entirely using "local" properties: a region is classified according to information supplied exclusively by its own vertices.

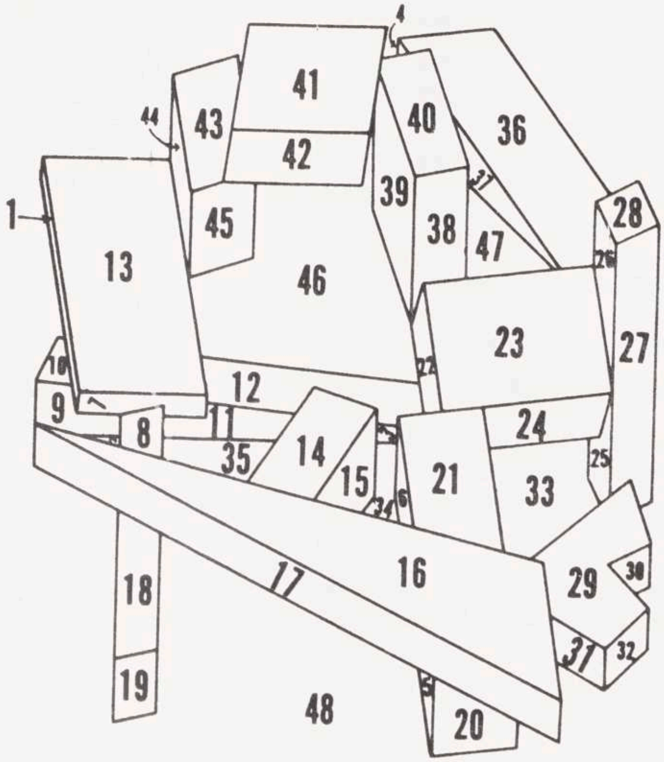


FIGURE 'S P R E A D'
Each region is classified as LIMPIO,
SUSPICIOUS or BACKGROUND.

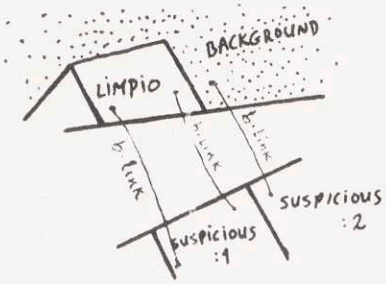
<u>More global indications</u> Our goal is to decide which of the suspicious regions are LIMPIO, and which ones are BACKGROUND.

== Since two background regions can not be contiguous (the background can not be neighbor of itself), suspicious regions that are contiguous with the background are cleaned and put in the LIMPIO status.

In our example, :48 is background and therefore its suspicious neighbor :18 gets cleaned and becomes LIMPIO.

== Links are established through the matching T's. We call them b'links.

Ideally, a suspicious region b-linked to a LIMPIO region gets cleaned, a suspicious region b-linked to the background gets converted to background too.



Idealizing, suspicious region :1 becomes LIMPIO, and suspicious region :2 becomes background. A more complicated procedure is actually used.

In practice, we allow for small errors as follows:

For each suspicious region, we notice if it is b·linked to background (BA), suspicious (SO), or Limpio (LI).

BA == == If it is b·linked to background regions, we change it to Background, except if it has a background as neighbor, in which case we do nothing and continue.

() SO LI If not b·linked to background, but b·linked both to Suspicious and Limpio regions,
(1) If LI < SO, continue, do nothing.
(2) If LI ≥ SO, classify this region as limpio (LI is the number of LIMPIO regions b·linked to the current region under consideration).

() SO () If b·linked only to suspicious, continue, do nothing.

() () LI If b·linked only to Limpio, change it to Limpio. Note: Sometimes I write Limpio, sometimes LIMPIO, they mean the same.

() () () If not b·linked, continue, do nothing.

We keep applying these rules until no change is observed. In this way, we have elliminated several suspicious regions.

In SPREAD, the suspicious regions were 35, 18, 34, 2, 3, 12, 11, 33, 37, 47, 48, 46. :48 is known to be the background (that was done in page 226), so is no longer suspicious. :18 is a neighbor of the background (:48), and got cleaned in the page before this one.

:11 is b·linked with the LIMPIO :9 and with the suspicious :3. Therefore, :11 changes to LIMPIO.

:3 is b·linked with the Limpio :11, so the suspicious :3 becomes Limpio.

:12 is b·linked to the Limpio :10, and gets cleaned.

:46 is b·linked to the background :48, and gets made background, since :46 is not, at this moment, neighbor of background.

:34 is b-linked to the background :48, and gets made background, since :34 is not neighbor of background.

:37 is b-linked to the LIMPIO region :4, and transforms into LIMPIO.

:35 is b-linked to the region :34, which is background, so that the suspicious region :35 becomes background instead. It is also b-linked to 48

:2 is a suspicious region b-linked to the region :35, which is part of the background. According to our rules, :2 becomes part of the background. :2 is also b-linked to the background :48.

At the end, only regions :33 and :47 remain suspicious:

(SUSPICIOUS ARE (:33 :47))

== We collect all these 'stubborn' suspicious regions and label them background, except those which are neighbors of background. A better procedure may be to make the exception in those regions that are neighbors of suspicious regions. That is, two neighboring suspicious regions prevent each other from becoming background. I have not explored this possibility. SUGGESTION

In the example SPREAD, :33 and :47 are made background.

== If no region is background at this point, make one of the "bigfaces" background. There is room here for improvement.

== If no background yet, make background the region with more vertices. Not yet implemented.

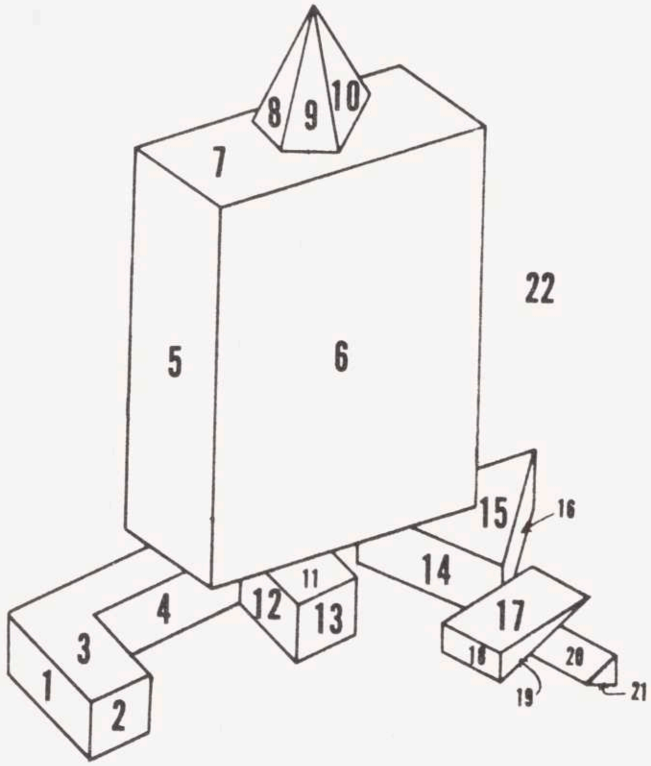
In our example, the (final) background regions are:

:33 :47 :35 :34 :2 :48 :46. ← BACKGROUND OF 'SPREAD'.

Other examples of background finding.

Scene CORN

```
LLENA
FOOP
SLOPGENERATOR
TYPEGENERATOR
MATES
NEXTE
SEARCHING FOR BACKGROUNDS OF CORN
(SUSPICIOUS ARE NIL)
     THE BACKGROUND OF CORN IS
(:22)
```



Scene BRIDGE

```
(:30 IS BIGFACE)
(SUSPICIOUS ARE NIL)
     THE BACKGROUND OF BRIDGE IS
(:30)
(:30)
```

Scene MOMO One mistake (:31) is produced here.

```
LLENA
FOOP
SLOPGENERATOR
TYPEGENERATOR
MATES
NEXTE
SEARCHING FOR BACKGROUNDS OF MOMO
(SUSPICIOUS ARE (:31))
     THE BACKGROUND OF MOMO IS
(:6 :31 :40)
```

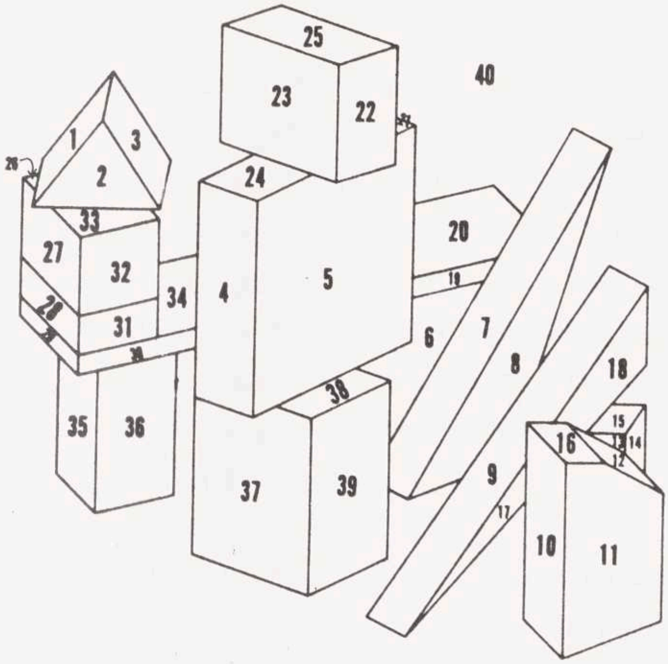


FIGURE —'MOMO.'

The problem is ambiguous Like in the case of body isolation (section 'The Concept of a Body'), the problem of determining the regions that belong to the background of a scene (regions that belong to no body) is ambiguous; many solutions are possible, as long as no two background regions are contiguous.

Among the multitude of solutions exists a preferred one, which is "the" standard (common, familiar) interpretation chosen by people.

Our program tries to choose also, among the many solutions, the standard one.

Summary A lenient algorithm finds regions (by analyzing the types of their vertices, and their neighborhood relations) that may possibly be background, and labels them "SUSPICIOUS". With the idea of re-classifying the suspicious regions as 'LIMPIO' (clean, no background) or 'BACKGROUND', a system of b·links is introduced. These b·links provide more global information about the scene.

Members of the suspicious set are assigned to one of the other two sets (limpio or background), while the algorithm tries to minimize the b·links between Background and Limpio regions.

Conclusion Fair results are obtained with the algorithm just described. Sometimes, regions are obtained as Background that are genuine components of a body ("Limpio") and viceversa.

Refinements are needed, but since in our present vision experiments the background is a homogeneous black area (see first few pictures of this thesis), no emphasis is shown right now.

STEREO PERCEPTION

Summary So far we have discussed the identification of objects in a scene and ignored the problem of locating them in a three-dimensional space.

There are several ways to achieve this. We will discuss here one of them: the use of more than one view of the same scene.

A natural first step is to establish the correspondence between points in the two views; that is, given a point in one scene (left), to find the corresponding point in the other scene (right). Theorems S-1 below and S-2 on page 234 express criteria for this "stereo matching".

> **THEOREM S-1**
>
> If both cameras are identical, their optical axes parallel and the films or sensitive surfaces or retinas lie in the same plane,
> then a simple necessary condition for two image points, one in each retina, to have come from the same 3-dim point, is that both image points (left and right) have the same y-coordinate,
> measured in the direction perpendicular to the line joining the optical centers.

SEE can independently decompose the left and right scene in the bodies forming them, leaving as a problem to determine which of the objects in the right scene corresponds to an object in the left scene. This can be done because each object will appear in both views with the same maximum height and minimum height (highest and lowest values of the y-coordinate of points belonging to that object); comparisons are easily made by replacing the objects by "intervals" consisting of these two numbers.

Further disambiguation can be achieved by the use of the function (WHERE X_L Y_L X_R Y_R), which determines the (x, y, z) 3-dim position of a point of which its two 2-dim locations (X_L, Y_L) and (X_R, Y_R) are known. {Griffith, AI Memo 143}.

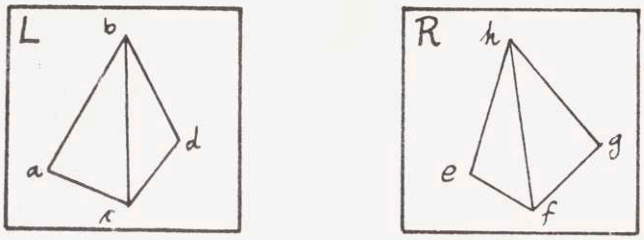


Figure 'P O I N T S'

Given two images of the same scene, before we can proceed to situate it in 3-dim space, it is necessary to know which points of the left scene correspond to points of the right scene: we have to discover the <u>genuine</u> <u>pairs</u> in it, a small subset of the cartesian product (a, b, c, d) $\times$ (e, f, g, h). It is desirable to have an algorithm that avoids an exhaustive search on this product.

<u>Genuine Pair</u> (definition). A pair of points (P_L, P_R) produced by a real 3-dim point of the scene in consideration.

Theorem S-2 below gives conditions that a genuine pair must meet. A particularization will produce theorem S-1 above.

THEOREM S-2

The left image P_L and the right image P_R of a point P have associated with them a variable, computable from (X_L, Y_L) or from (X_R, Y_R), that will acquire the same value on P_L and on P_R. It is invariant under change of scene.

For the case where the optical axes are parallel, this variable is simply the y-coordinate ($Y_L = Y_R$) or height of the image.

For the case where the optical axes meet, this variable is γ, an angle that plane P_L-C_L-P-C_R-P_R makes with Γ, the plane containing the optical axes.

Any monotonic function of γ will be just as good. (cf. figure 'GENUINE PAIRS').

From the theorem, the algorithm (referred to in fig. 'POINTS') that we may use to establish correspondence between points in the two views is:

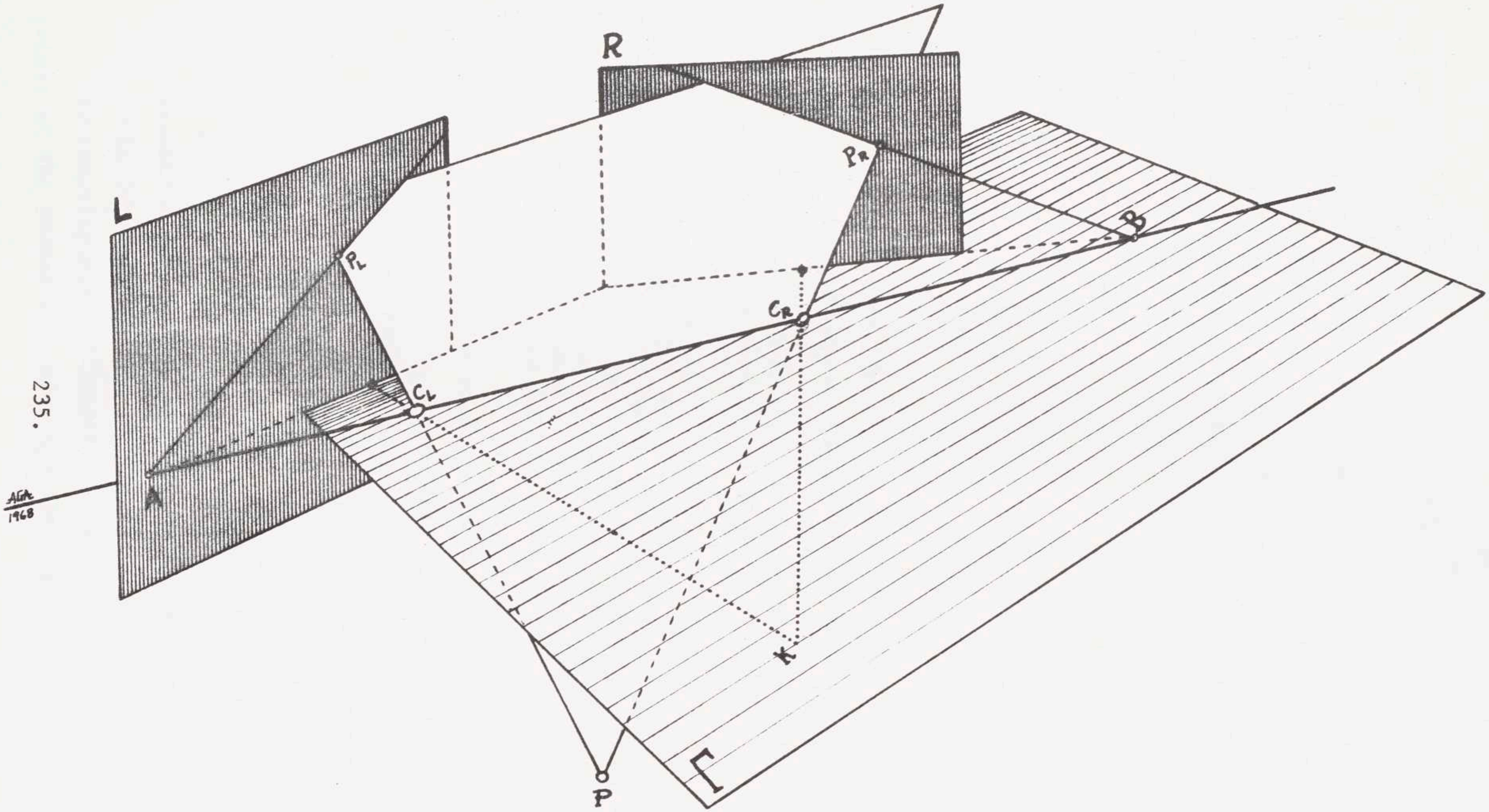


FIGURE 'G E N U I N E   P A I R S'

Compare only points with the same γ (or the same y-coordinate).
Points with different γ can not come from a genuine pair.

For each body, the knowledge of the 3-dim location of a few of its vertices will be sufficient to position that body in real space, achieving in this way the goal of this section.

See Digression 1 in section 'The concept of a body', for a different approach.

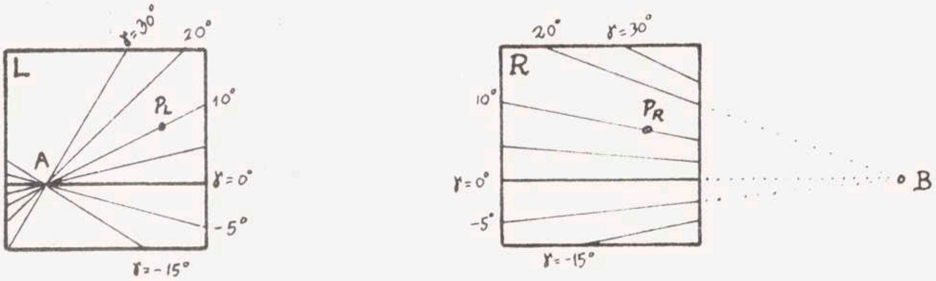


Figure 'γ - P A R A M E T R I Z A T I O N'

From geometrical considerations and the coordinates of a point P_L in L, it is possible to attach to the line A-P_L an angle γ. Similarly, an angle is obtained for lines of R. It can now be said that a genuine pair (P_L, P_R) must have the same γ's for P_L and P_R.

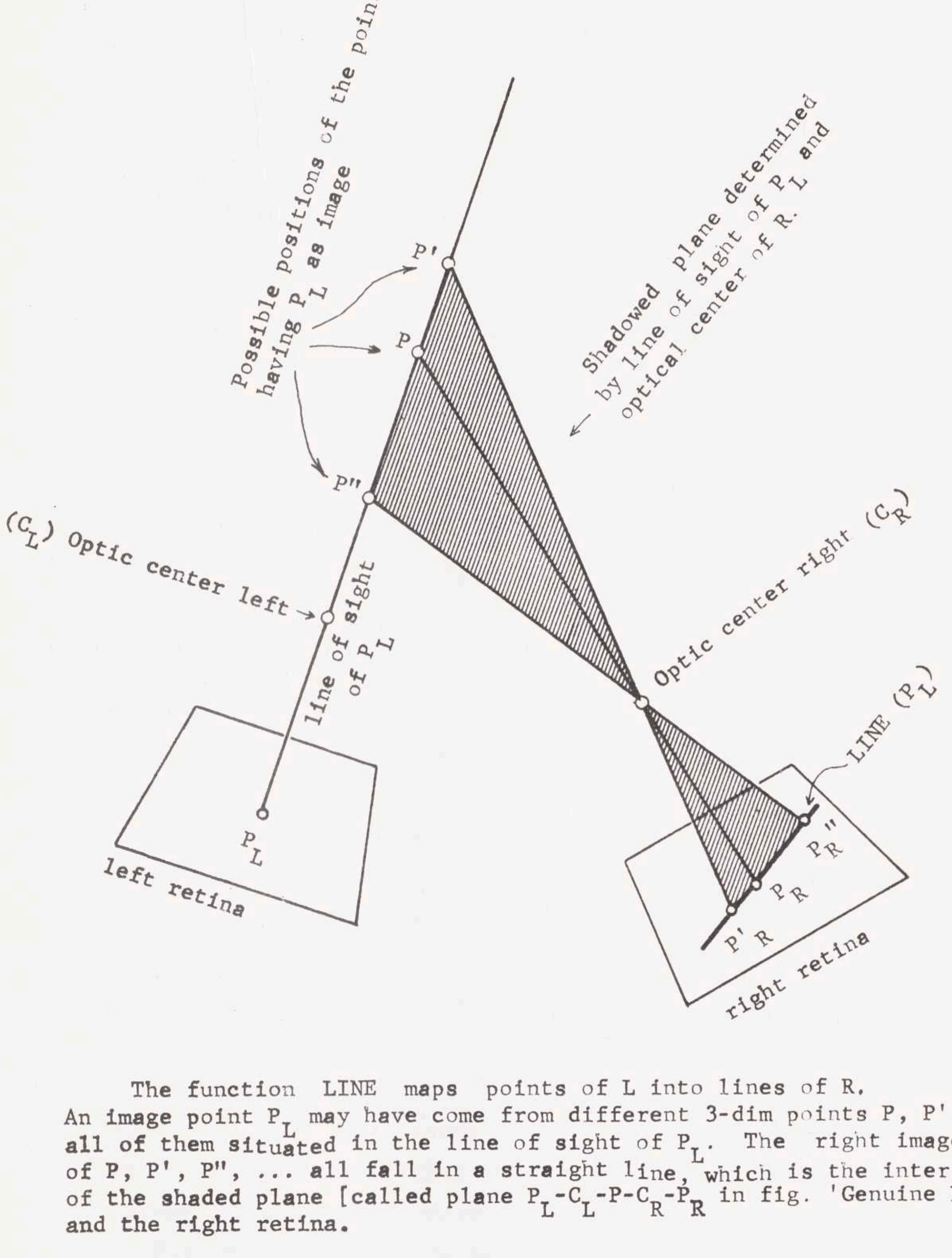
γ is a physical quantity, namely the angle that the plane passing by the image P_L and the optical centers C_L and C_R makes with the "horizontal" plane Γ. (Γ contains the optical axes). Clearly, for P_L and P_R to be produced by a point P in 3-dim space, the γ of P_L must be equal to the γ of P_R. This is a necessary condition that is easy to check.

A real point P of the scene produces a left image P_L (which has a certain value of γ) and a right image P_R with the same value of γ (figure 'γ-PARAMETRIZATION').

Thus, given a point in one scene, we have to search for its genuine pairs in the other scene among the points with its same γ. They will be found along an straight line through A or B.

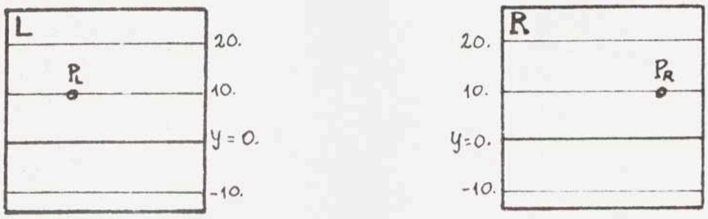
Parametrization of the scene is possible not only by using γ; a monotonic function of γ will do.

For computational efficiency, it may be advisable to store the points of the scenes into arrays according to the value of their γ's.



The function LINE maps points of L into lines of R. An image point P_L may have come from different 3-dim points P, P', P"... all of them situated in the line of sight of P_L. The right images of P, P', P", ... all fall in a straight line, which is the intersection of the shaded plane [called plane P_L-C_L-P-C_R-P_R in fig. 'Genuine Pairs'] and the right retina.

When the optical axes are parallel In this case, points A and B on line C_L-C_R (fig. 'Genuine Pairs') travel to infinity, and lines P_L-A and P_R-B become horizontal (parallel to C_L-C_R). The situation looks like



A genuine pair (P_L, P_R) will have the same y-coordinate for both of its elements (10.0 in this case).

So that, given a left image point P_L, we have to search only among the points of R with its same height, to find "the" P_R that will make a genuine pair (P_L, P_E).

But several genuine pairs may be found. Because on each horizontal line on R, many points may lie.

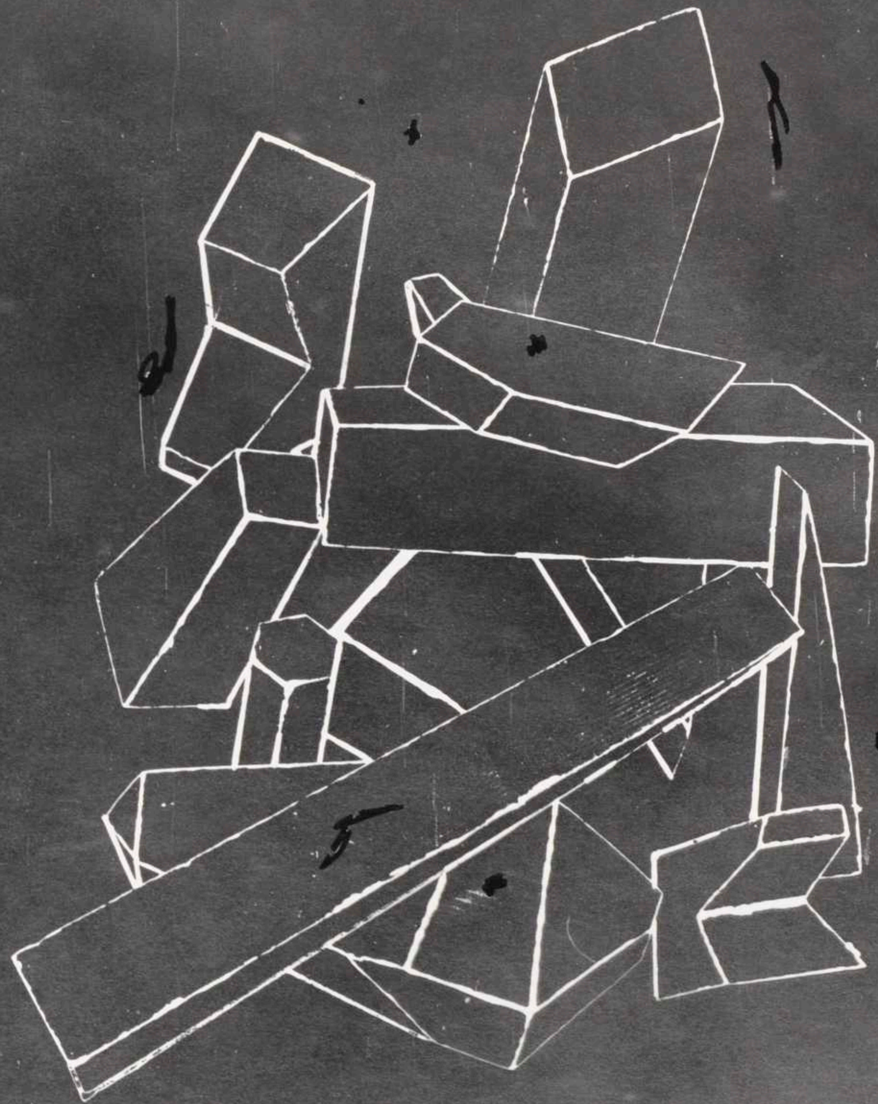
USE OF SEE IN STEREO PERCEPTION

We can use the invariance of the variable described in Theorem S-2 to locate objects in three dimensional space, from a pair of stereo views (we will suppose parallel axes; other case is similarly treated), as follows:

(1) Make an analysis of the left scene with SEE, identifying the bodies.
(2) Id. for right scene.
(3) Reduce each body to an interval formed by two numbers, its maximum and minimum height, specifying "closed" if the absolute extremal of the body is known, "open" if not.
In this way we reduce each scene to a set of intervals (see figure 'INTERVALS').

FIGURE 'L 1 3'

This picture and the one in next page represent a stereo pair, taken with cameras having parallel optical axes. See fig. INTERVALS.



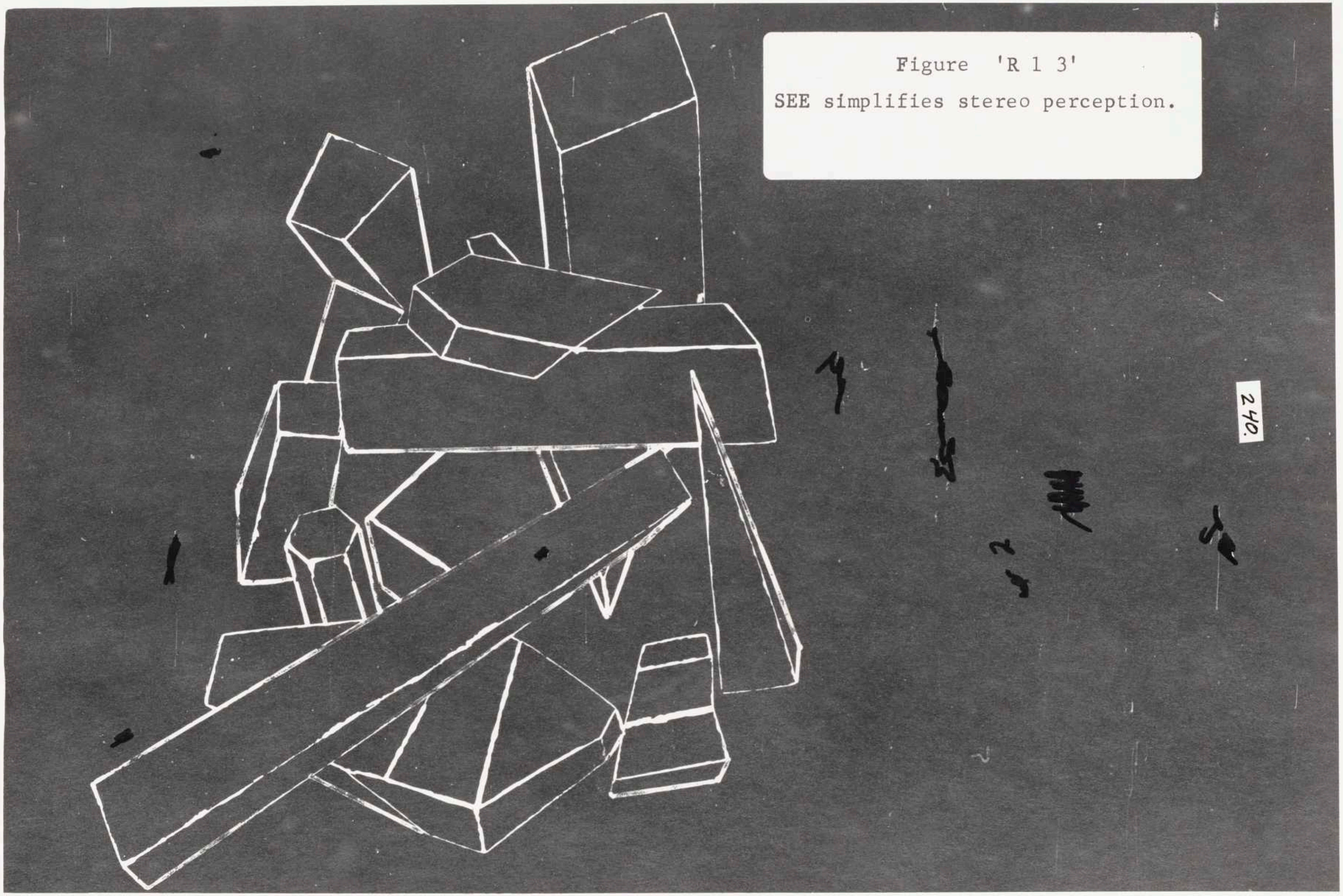
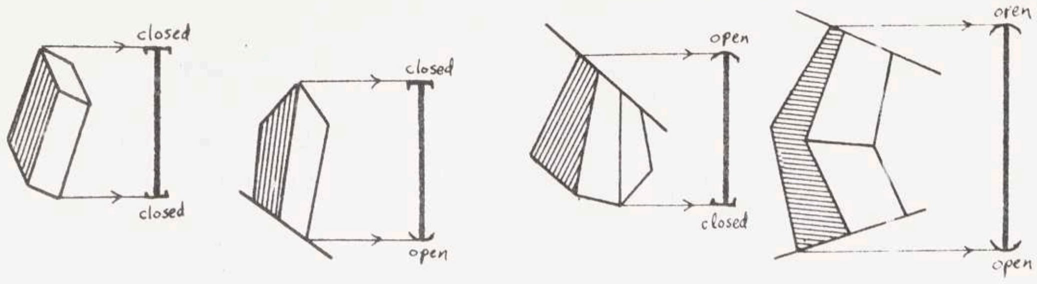


Figure 'R 1 3'
SEE simplifies stereo perception.

240.



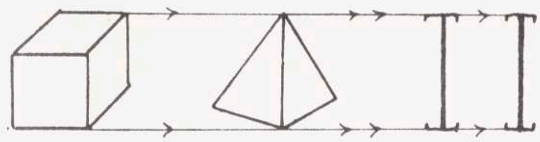
Each body is reduced
to an interval.

(4) Use these intervals to select which left body will go with what right body. The answer is simple (because it is unique) even in moderately crowded scenes.

It is simple to take into account the fact that an open end of an interval indicates that the interval can extend further at such end.

Sources of difficulties are:

(a) Two bodies have the same interval, meaning they have identical maximum heights and minimum heights. This is possible.



Quite easy: reduce some faces to intervals and compare them.

(b) A body is seen in left scene but not in right scene (figures L12, R12).

(c) SEE partitions one body in two in one scene, but not in the other.

The "open" and "close" indications will help here.

Also, remember that we are using, when comparing these intervals, just a very small part of the total information concerning each body. When the selection is narrowed down to two or three candidates ["left-body 1 is either right-body 2 or right-body 5 "], one can use

(1) the WHERE function of Griffith (op cit),

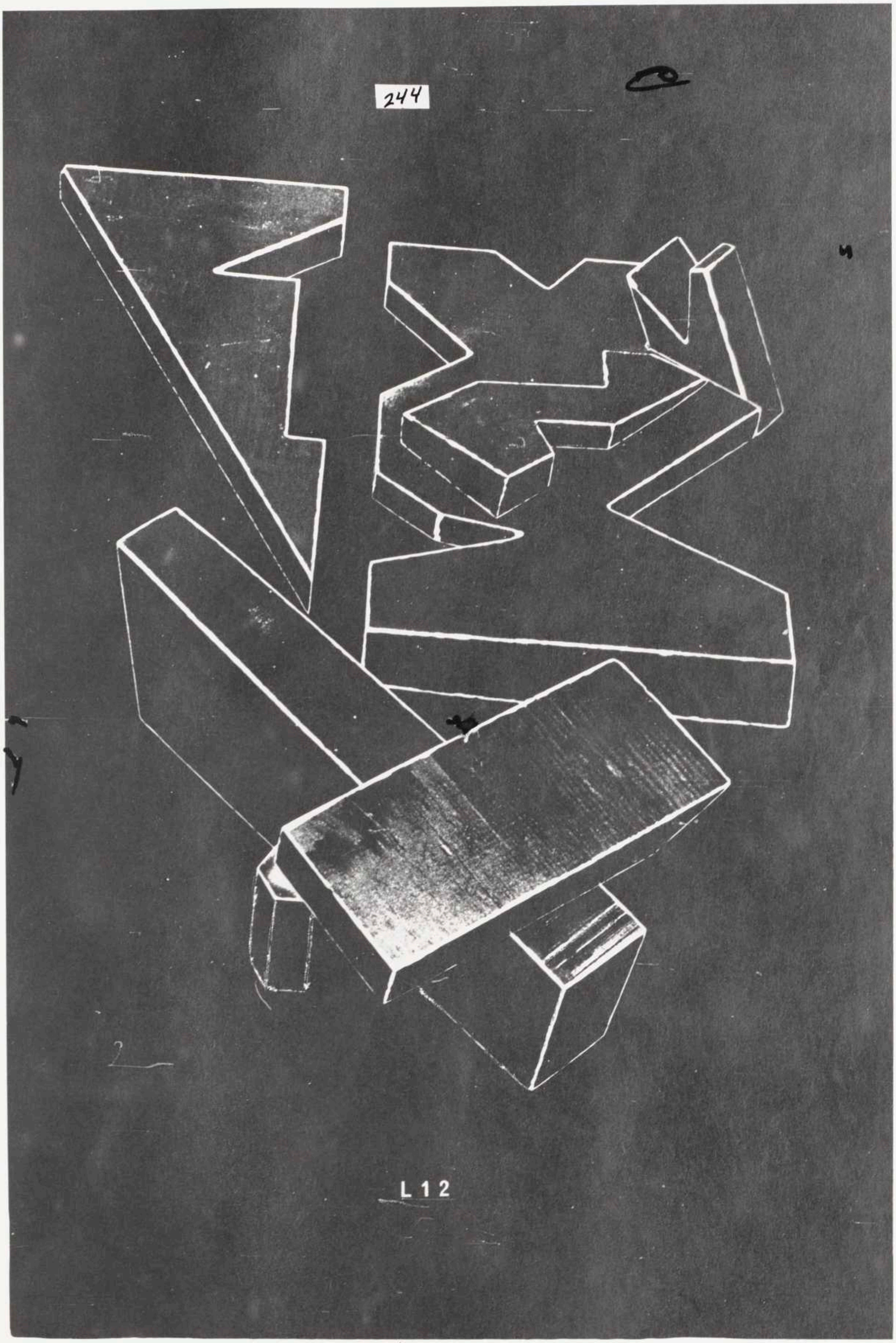
(2) as in (a) above, the intervals for each face of the objects, so as to chose as "genuine pair" those two objects with more agreement in the intervals of their faces;

(3) perhaps a face of unusual shape is enough for discrimination, if it appears both in left and right scenes, or the number of vertices below the center of gravity, or ...

summary

In summary, I should like to point out that, while much has been stated within the somewhat constricting framework of this article, much remains to be stated. Certain, but not all, important classes of presentations have been treated, and there remain horizons as yet unexplored. Conceivably, the author will attempt, *ex nihilo nihil fit*, to establish a more general perspective in the course of a subsequent article. (D.M. Jones, Datamation Nov 68). ■

Also, the reader is referred to other articles on the same topic.



L12

FIGURE "R 1 2"

For the pair L12 - R12, caution should be exercised, because an hexagonal prism disappears from L12 and a brick appears in R12.

Scene L10 - R10 SEE analyzes independently (pages 125 and 128) the left and right scenes, obtaining the following bodies:

```
(BODY 1. IS :5 :1 :4 :12)               LEFT SCENE (L10)
(BODY 2. IS :6 :15 :7 :11 :14)
(BODY 3. IS :8 :9 :10 :3)
(BODY 4. IS :2 :13)

                   (BODY 1. IS %:3 %:5 %:6 %:14)
RIGHT SCENE (R10)  (BODY 2. IS %:13 %:1 %:11 %:9 %:15)
                   (BODY 3. IS %:8 %:2 %:10)
                   (BODY 4. IS %:4 %:7 %:12)
```

For each of the eight bodies, we compute its minimum height and its maximum height, obtaining the following intervals:

L10		R10	
:5 :1 :4 :12 →	[66,105)	[67,154] ←	%:3 %:5 %:6 %:14
:6 :15 :7 :11 :14 →	[79,120]	[78,119]	%:13 %:1 %:11 %:9 %:15
:8 :9 :10 :3 →	[68,152]	[65,103) ←	%:8 %:2 %:10
:2 :13 →	[21,82)	[22,82) ←	%:4 %:7 %:12

These intervals are compared (left with right), trying to find pairs with discrepancies between their values tolerably small [if the interval has an open end, differences can be larger]. For 'L10 - R10', these are

[66,105) = [65,103)

[79,120] = [78,119]

[68,152] = [67,154]

[21,82) = [22,82)

that corresponds to the following identification of bodies:

:5 :1 :4 :12 corresponds to %:8 %:2 %:10

:6 :15 :7 :11 :14 corresponds to %:13 %:1 %:11 %:9 %:15

:8 :9 :10 :3 corresponds to %:3 %:5 %:6 %:14

:2 :13 corresponds to %:4 %:7 %:12

Once these correspondences between objects in the two images are found, the function (WHERE ...) {Griffith} will position these bodies in three-dimensional space, achieving our goal.

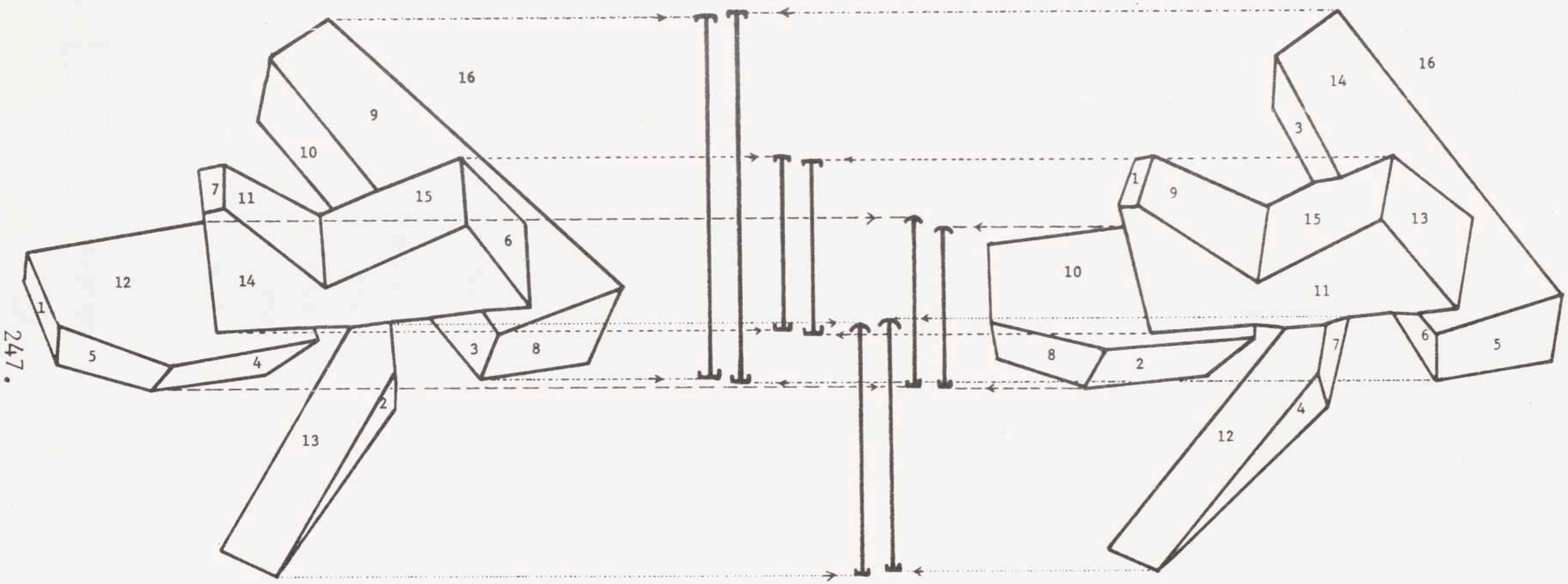


FIGURE 'L10-R10'

In order to find the three-dimensional position of these bodies in space, we proceed as follows: First, SEE analyzes the left scene, and finds (according to page 125, 'RESULTS FOR L10') four bodies: 5-1-4-12, 6-15-7-11-14, 8-9-10-3, and 2-13. Then, SEE analyzes the right scene (see 'RESULTS FOR R10' in page 128), and finds the following four bodies: %3-%5- %6-%14, %13-%11-%1-%9-%15, %8-%2-%10, and %4-%7-%12. Then, each one of these bodies is reduced to intervals consisting of two numbers (represented by heavy vertical lines in this drawing), the maximum and minimum height. Left and right intervals are compared and grouped in pairs consisting of intervals with the least discrepancy in their values at the ends. These pairs indicate the matching or correspondence among bodies. For this pair of scenes, the correspondence is: 5-1-4-12 with %8-%2-%10, 6-15-7-11-14 with %13-%11-%1-%9-%15, 8-9-10-3 with %3-%5-%6-%14, and 2-13 with %4-%7-%12. Once this correspondence is known, geometric considerations alone {see Griffith, AI Memo 143} permit us to find their three-dimensional positions.

C O N C L U S I O N S

LOOKING BEHIND

When I started to work on these problems, the idea was to describe an object by using a model, and with this model in memory, to search the scene looking for sub-parts of it that would fit the description.

This work ended (as far as this thesis is concerned) with a program that finds bodies without having a model of them.

But that is good.

We did not know at the beginning that this could be done.

LOOKING AHEAD

a. Suggestions for further work
b. Comments
c. Recommendations
d. Summary
e. Conclusions
f. Evaluation
g. Extensions and Implications

All these matters are normally encountered grouped in a chapter at the end of the work

I can only partially lump all these important matters in one final section; many times I cite them in context, that is, next to the figure or subject that evokes them, or with which they are most closely related. As a result, they are spread through the body of this dissertation.

Also,

(1) The box SUGGESTION appears through this thesis near a

partially unsolved or partially formulated problem, and/or its partially outlined or partially new solution.

(2) In page 256 there is a list of such suggestion boxes.

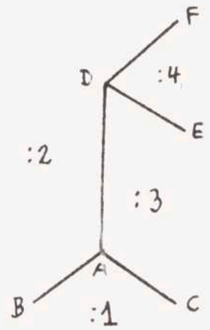
(3) The remaining portion of this section and, in general, the sections close to the end of this work, abound in statements of type (a.) through (g.).

(4) I have tried to start each section with a brief, and end it with a summary or conclusion.

(5) The section 'Introduction' (page 10) specifies the problems treated in this thesis, and the section 'Preliminary view of Scene Analysis' (page 14) produces a general view of available methods.

General notation To put, remove, etc., links, we may develop a notation that will look like

SUGGESTION




```
(WHEN  A  (Y A)  (B :1 C :3 D :2)
          D  (K ( A F ..))  (A :3 E :4 F :2)
   THEN
          PUT LINK   KIND 3   :3  :4
          NO LINK   :1  :2     )
```

"When A is a vertex of type 'Y', and
D is a vertex of type 'K', and
A and D are joined as specified,
then
put a link of kind 3 between region :3 and :4, and
do not put a link between :2 and :1."

The general notation is

(WHEN P E E')

"when predicate P is satisfied, evaluate expression E (execute E), otherwise execute E' (which may be missing)".

In this notation, the predicate P corresponds to a geometric pattern or configuration, and the expressions E and E' to the establishment or removal of links.

In SEE, this part is handled by LISP functions (hand-coded), one for each particular heuristic. The suggestion is to develop this general notation, and an interpreter for it. This will speed up programming and checking, but will slow down to some extent the execution.

Use The main use of the new notation or language is for trying new heuristics. Actually, it is not difficult to hand-code the new heuristic in LISP (see function EVERTICES in listings), because everything reduces to calls to NOSABO, THROUGHTES, GEV, SUME, etc. I was thinking that a simple MACRO of Lisp could transform from notation (WHEN P E E') to Lisp functional calls.

Since what the notation or language is really doing is expressing as a linear string a two-dimensional configuration  , a more ambitious project would be to use the light pen and draw this configuration, and then have our interpreter or compiler produce the Lisp program. This may look a little like AMBIT-G {Christensen}.

Assigning a name to an object

Problem. SEE has separated a scene into bodies. What are they? Is there a pyramid among them? Where are the parallelepipeds?

To answer this, information can be supplied to the program, in the form of a symbolic description or model of the object we are trying to find. A model is an idealized account of a class of objects, all receiving the same name, like "triangular pyramid" or "house". Models may have parameters that acquire values after a given instance of the model has been found in a scene. Examples are "height" or "length of bottom side".

Some programs that follow the above procedure to name objects in a scene are described and discussed in a Master's Thesis {Guzmán}. There are difficult problems to be solved if we are to make the system able to recognize occluded objects in many situations.

One could, of course, bypass SEE and look for particular objects, as it is done by Polybrick {Hawaii 69}, a program that finds parallelepipeds.

Do not use over·specialized assumptions. Use more information In trying to solve a problem, people will apply quite different methods. They may also suppose quite different assumptions, some of which may not hold. Due to particular experience, environment, preferences, etc., some subjects may be using over·specialized assumptions, instead of requesting more data, more information to solve the problem. We may bias our views and risk arriving at conclusions (of the "common sense" type) which are valid only on restricted segments of populations, or in particular conditions or situations.

Holes. For instance, if most of the readers of this thesis [technical specialists, who have learned to read, are interested in graphical processing and computers, etc; who may not be considered a representative cross-section of Homo Sapiens] perceive "objects" a, b and c of figure 'HOLES' as holes {Winston}, we may be tempted to conclude that this is a general property, and rush to write a

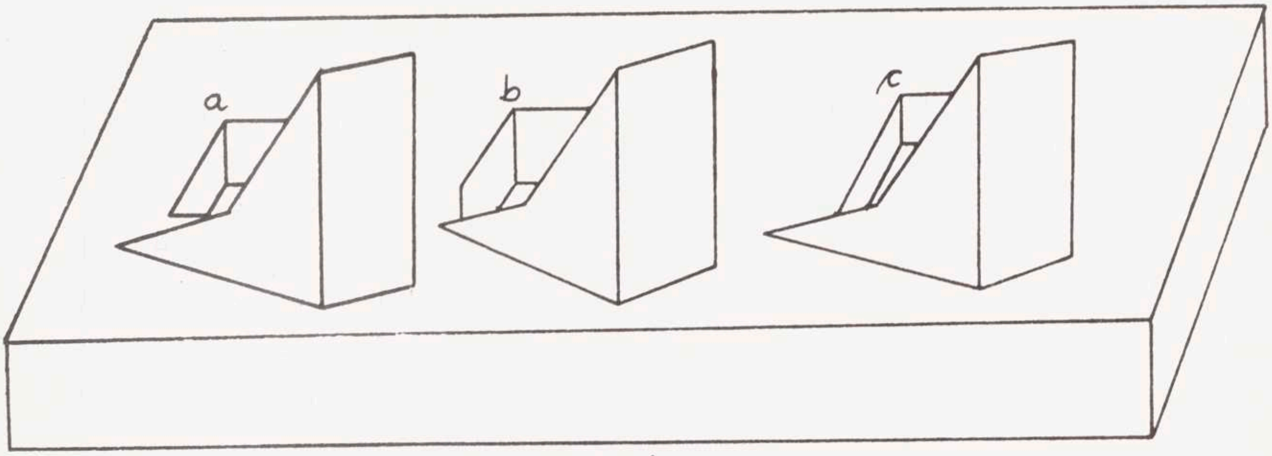


Fig. 'HOLES'

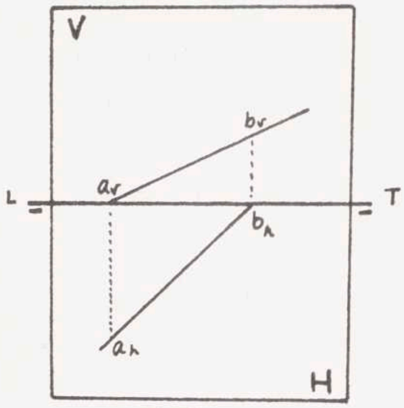
The *idea* that objects a, b, c have to be interpreted by all men, and hence by a program, as holes in the larger box, is dangerous. {cf. AI Memo 163}

subroutine to find such orifices. Perhaps other sectors of our population would simply say, with respect to a, b, c, of figure 'HOLES' that "there is not enough information to make a decision" (see also section 'On optical illusions'). Or they may come with different answers, using <u>their</u> set of assumptions which may be different from ours, since their experience is different too. The Ames' Room (see Box, page 201) and Gregory (see Box) warn us of this.

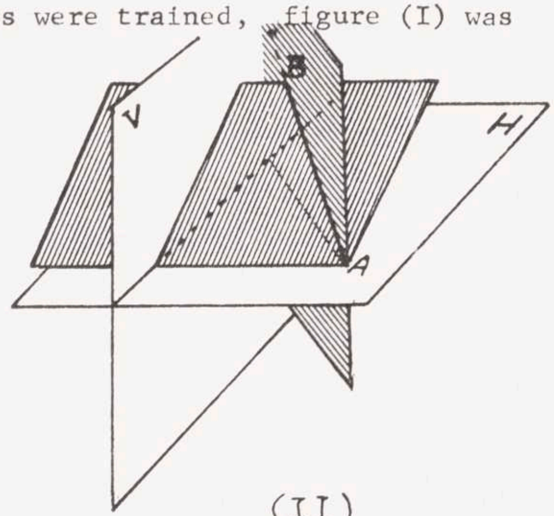
<u>Other example of over-specialization</u> For people familiar with Descriptive Geometry, it is easy to see that figure 'DESCRIPTIVE' (I) shows a straight line in the first octant. For them, indeed, it is easy to visualize this line in three dimensions and have a fairly good idea of its position and orientation in space, just from figure (I).

Other persons would need a more conventional figure, such as figure 'DESCRIPTIVE' (II), to visualize the same line, to get the same idea.

What happened was that the first group of persons were using especialized knowledge, their eyes were trained, figure (I) was familiar to them, etc.



(I)



(II)

Figure 'DESCRIPTIVE'

<u>Conclusion</u> Before looking for heuristics and shortcuts, before making assumptions, deductions, etc., let us be sure that there is enough data to solve our problem.

Human perception versus computer perception Given a two-dimensional line-drawing of a three-dimensional scene, the problem of finding bodies in it is inherently ambiguous: many 3-dim scenes can generate the same 2-dim scene.

Multiple solutions are possible. More over, the metatheorem of page 39 guarantees that a solution always exists, and provides ways to construct it. We call this solution "trivial"; in effect it is trivial to write a computer program that will invariably find it.

From the multitude of possible solutions, human beings select one, which is * different from the trivial, and call it "normal" or "common" or "standard" or "reasonable" interpretation of the scene.

Our program SEE also selects one of the many solutions. How does its selection compare with the human choice?

== When the scene is "clear", in the sense of evoking human unanimity, SEE will * also select that same answer. Example: Figure 'TOWER'.

== As the scene or drawing gets complicated or ambiguous, mortal behavior deteriorates; opinions split, optical illusions may emerge (indicating contradictory evidence perceived), several plausible answers are emited.

The answer of SEE in these cases will * be found among the humanly plausible selections. In some cases, it may not agree with the majority.

== Finally, people make mistakes. They will see an object that is not there, or will fail to see an object, or classify it as "impossible".

SEE also errs. It sometimes succeeds where people fail, and more often is the other way around.

---

* In an overwhelming majority of cases.

T A B L E "A S S U M P T I O N S"

ASSUMPTIONS MADE BY THE PROGRAM

These assumptions have to be obeyed for SEE to give good results:

== The objects are three-dimensional solids formed by planes [(1)]. No needles or cardboards allowed.

== They produce a two-dimensional image or projection where all lines are straight[(2)].

== Faces have no drawings, marks, labels, etc., imprinted on.

== Objects do not have holes in them.

---

[1] See section 'On optical illusions' for conditions for partial lifting of this assumption.

[2] See section 'On curved objects' for conditions for partial lifting of this assumption.

ASSUMPTIONS NOT MADE BY THE PROGRAM

These assumptions are not necessary for the correct functioning of SEE; it will work well with or without them.

== Only prisms are allowed.

== The scene is a parallel projection, or isometric drawing.

== The objects are convex.

== The model or description of the object has to be known to SEE.

== The objects have to appear unoccluded or unobstructed in the view.

== The objects have "weight" in the vertical direction and will fall if not supported.

== The background is known in advance (See 'On background discrimination by computer').

I repeat, these assumptions are NOT obeyed by our program.

LIST OF SUGGESTIONS

	PAGE
	75
	85
	88
	107
	107
	146
	173
	183
	187
	200
	205
	221
	229
	248
	250

R E F E R E N C E S

All Project MAC memoranda (such as MAC M 357), technical reports (such as MAC TR 37), and Artificial Intelligence memoranda (such as AI Memo 130), are internal working papers of Project MAC, a research project of the Massachusetts Institute of Technology, and may be obtained from their respective authors at the following address:

Project MAC, M. I. T.
545 Technology Square
Cambridge, Massachusetts. USA. 02139

This is also the present address of the author.

Blum, Manuel. A Line proposer. Unpublished memorandum and description. 1968.

Canaday, R. H. The Description of Overlapping Figures. M. S. Thesis, Electrical Engineering Dept., M. I. T. 1962.

Christensen, Carlos. An Example of the Manipulation of Directed Graphs in the AMBIT/G Programming Language. Proceedings of the Symposium on Interactive Systems for Experimental applied Mathematics. Washington, D.C. August 26-28,1967. Also as: Paper CA-6711-1511; Computer Associates, Inc. Wakefield, Mass.

== On the implementation of AMBIT, a language for symbol manipulation Comm. ACM 9 (August 66), 570-573.

Conrad, D. Lisp functions for Generating and Plotting Figures. Program Note No. 2, Centro Nacional de Calculo, IPN, México.1964.

Evans, T. G. A description-controlled pattern analyzer. Proceedings of the IFIP Congress Edinburgh 1968.

Freeman, H. Techniques for the digital computer analysis of chain-en coded arbitrary plane curves, Proc. National Electronics Conf., Chicago, Illinois, vol. 17, 421-432, October 9-11, 1961.

Grasselli, A. (ed) Proceedings of the Conference on Automatic Interpretation and Classification of Images (A NATO Advanced Study Institute), Pisa, Italy, Aug 26-Sept 7, 1968. Academic Press, Inc. (to appear).

Gregory, R. L. Visual Illusions. Scientific American 219, 5, 66-76, November 1968.

== Visual Illusions. pp 68-96 of New Horizons in Psychology, B. M. Foss (ed) Penguin Books (England), 1966.

== Will seeing machines have illusions? pp 169-177 of Machine Intelligence 1, N. L. Collins and D. Michie (eds). Oliver and Boyd, Edinburgh and London, 1967.

Griffith, A. K. in AI Memo 143 {Minsky}.

Guzmán, A. (see "Publications and technical reports," pages 286-287).

Hawkinson, L. Hawkinson-Yates Lisp for the IBM 709. México, 1964. (unpublished).

Jones, D. M. Presenting papers for pleasure and profit. Datamation 14, 11, November 1968. p. 91.

Kain, R. Y. Aesop and the Referee: A Fable. Comm. ACM 10, 3, March 1967, p. 138 (letter).

Kirsch, R. A. Computer interpretation of English text and picture patterns. IEEE Transactions on Electronic Computers EC-13 4 363-376 August 1964

Krech, D., and Crutchfield, R. S. Elements of Psychology. A.A.Knopf, New York, 1958.

Ledley, R. S. Programming and utilizing digital computers. McGraw-Hill New York 1962 (chapter 8)

==, and Wilson, J. B. Automatic Programming Languages translation through syntactical analysis. Comm. ACM 5, 3 March 1962. 145-155

== , Rotolo, Golab, Jacobsen, Ginsberg, and Wilson. FIDAC: Film input to digital automatic computer and associated syntax-directed pattern-recognition programming system. Optical and Electro-Optical Information Processing J. Tippett et al (eds). MIT Press Cambridge, Mass 1965 Chapter 33.

[11]McCarthy, J., Reddy, D., Earnest, L., and Vicens, P. J. A Computer with hands, eyes and ears. Proceedings of the AFIPS FJCC Vol 33, 329-338, December 1968. Thompson Book Co.

McIntosh, H. V. A Handbook of Lisp Functions. RIAS Technical Report 61-11. Baltimore, Md. 1961.
MBLISP is described in An Introduction to Lisp, A. K. Griffith, University of Florida, 1962.

McIntosh, H. V., and Guzmán, A. A Miscellany of CONVERT programming. Project MAC Memorandum MAC M 346 (AI Memo 130). April 1967.

Minsky, M. Stereo and Perspective Calculations. AI Memo 143. Sept 67.

[10]==, and Papert, S. A. Research on intelligent automata. Status Report II, Project MAC, MIT September 1967.

Narasimhan, R. A linguistic approach to pattern recognition. Report No.121, Digital Computer Laboratory, Univ. of Illinois. July 62.

== Syntax-directed interpretation of classes of pictures. Comm.ACM 9, 3 166-173. March 1966.

== Labeling schemata and syntactic description of pictures. Information and Control 7, 151-179. 1964.

Project MAC Progress Report IV. July 1966-July 1967.
Progress Report V. In preparation.

[12] Raphael, B. Programming a Robot. Proc. IFIPS Edinburgh 1968. See also {Rosen and Nilsson}.

Roberts, L. G. Machine perception of three-dimensional solids. Optical and Electrooptical information processing. pp 159-197. J T Tippett et al (eds) MIT Press 1965.

Rosen, C. A., and Nilsson, N. J. (eds) Application of Intelligent automata to reconnaissance. Third interim report Stanford Research Institute December 1967.

Shaw, A. C. The formal description and parsing of pictures. Ph D Thesis Computer Science Dept. Stanford University. Also issued as Technical Report SLCA 84 (Stanford Linear Accelerator Center) April 1968.

==, and Miller, W. F. Linguistic methods in picture processing -- a survey. Proc. AFIPS FJCC Vol 33, pp 279-290. Dec 68. Thompson Book Co. Washington, D.C.

Segovia, R. CONVERT en el diseño de procesadores. Professional Thesis (B. S., Electr. Eng.), ESIME, Inst. Politécnico Nacional, México, 1967. (Spanish)

Winston, P. H. Holes. AI Memo 163, Project MAC, MIT. August 1968.

Notes: Comm.ACM = Communications of the Association for Computing Machinery.

Proc. FJCC = Proceedings of the Fall Joint Computer Conference

Proc. SJCC = Id. Spring. (Spartan Books, or Thompson Books, Co. Washington, D. C.)

ANNOTATED LISTING OF THE FUNCTIONS USED

You do not have to know these things in order to use SEE (reading 'How to use the program' in page 78 is enough) or to understand what it does (it is explained in 'SEE, a program that finds bodies in a scene', page 58); these things are put here merely for completeness and to make easier the understanding of the inner workings of SEE.

A listing is a formal description There is a stronger reason, however. A listing of the programs is a formal description, an algorithm, an exact statement in a formal language of what we may have been describing, perhaps inaccurately, in a natural language (English). It becomes the starting point of serious discussions. The reader who is skeptical at some point, or did not understand some English statement, can always clarify his doubts in the listing. To be understandable, the listing has to have annotations, comments.

A mathematician is not forced to explain his work always in natural language, but rather he is allowed to employ abstract notations, symbolisms, formalizations of his thoughts (indeed, it is preferable this way). A programmer should not hide his listings (he should not be forced to re·state his algorithms in natural language exclusively { 68}) and force his readers to use the ambiguous channels of his natural language communication.

And this brings another point. Not only a programmer should not hide the listing (unless there are known bugs or incomplete subroutines), but he should not hide the programs (unless they are banal); by this I mean honest and reasonable efforts should be made to facilitate future potential users the access to these programs. Include:

- == Documentation
- == Listings, tape or card deck names, etc.
- == Test data
- == Printout of an interaction with such test data, including loading, compilation, execution, results.
- == Time spent (by machine and by man).

See also R. Kain's letter {C. ACM March 67}.

FILE DIRECTORY OF TAPE GUZMAN $

23 DICIEMBRE 1968 Adolfo Guzmán Arenas

GUS
FREE FILES 16 FREE BLOCKS 188

File		Blocks	Notes
BACK	MACRO	2	irrelevant
.TECO.	OUTPUT	0	irrelevant
GUFTPE	23DC68	348	← This file contains a backup of all the English (ASCII) files of Tape GUZMAN F, as of DEC 23, 1968 (this tape is shown below.)
DO	3	8	irrelevant
SQRT	LAP2	10	irrelevant
M.	TECO	3	irrelevant

FILE DIRECTORY OF TAPE GUZMAN F. ← It contains the programs and results. Data are in tapes GUZMAN C and GUZMAN CVT.

GUF
FREE FILES 7 FREE BLOCKS 46

File		Blocks	Notes
RESUL2	S58,57	40	Results of SEE on scenes L4, R17, L2, R3, L?, R19, L3, R2, L17, L17T, L19, R10.
SEECMP	57OLD	40	old
DISPLY	CUBE7	9	← Used to display a scene, or to draw it on the Calcomp Plotter.
RESUL3	S58,57	20	Results of SEE applied to Tower, Corn, Hard, Wrist.
SEE	1	160	SEE 1 is a binary dump of LISP (44 blocks) + All "SEE" programs. To load it: ↑Z $L SEE 1↵ $G — The file SEE 1 is not contained in the backup file GUFTPE 23DC68 $
SEECMP	57	40	One of the two files needed to create SEE. To load it into LISP: (UREAD SEECMP 57) ↑Q ↑Q ↑Q Disregard error messages. ← Use three ↑Q's
RESUL1	S58,57	14	Results of SEE applied to scenes R9, R9T, R4, L10.
EXFORM	TSCER3	7	Example of 'Format of A scene' or Internal Format of scene R3.
SEE	58	36	Load this file by (UREAD SEE /58) ↑Q into LISP, also load SQRT LAP2 and SEECMP 57 ← Use about 3,000 words of fullword space
RESULT	S58,53	41	Results of scenes Bridge, Spread, L9, Momo, Home, Wrist, Arch; They were obtained with SEE 58 and SEECMP 53
SQRT	LAP2	10	If you do not have a SQRT function, load this file. At the end, say (REMLAP T)
TRAC	.11	10	Tracing LISP functions.
RESUL5	S58,57	31	Results of SEE 58 and SEECMP 57 applied to scenes spread, momo, bridge, II (or stack), corn, home, tower.
RES4AR	CH,WRS	11	Results of ARCH and WRIST*.
SEE	59	36	Newest version of SEE as of DEC 23 1968.
RES6TR	L,HARD	8	Results of SEE on "TRIAL" and "HARD". All these results appear in this thesis in section 'Analysis of many scenes.'

In the next listing we show the files SEE 58 and SEECMP 57, which contain all the functions needed (except SQRT) to create "SEE". They are written in the LISP programming language.

TO USE THE PROGRAM, SEE PAGE 78

SEE 58 F   In tape GUZMAN F.

```
(SETQ PRONA (QUOTE (SEE /58)))
```

Version number 58.

R is a region. (suspicious R) = T if R is a candidate to be background, () otherwise.

```
(DEFPROP SUSPICIOUS (LAMBDA (R)
  (PROG (LS FORKS ARROWS TS KS XS PEAKS MULTIS VA NA L)
        (SETQ L (LENGTH (SETQ ** (GETQ R KVERTICES))))
        (CUENTA **)
        (COND (SB (BREAK WITH R LS FORKS ARROWS)))
        (COND (NA (PRINT (LIST NA * (QUOTE (ERROR SUSPICIOUS)))))
              ((GREATERP
                 (PLUS (LENGTH (HAVING (FUNCTION (LAMBDA (K) (EQ R (CAADR (GETQ K TYPE))))) LS))
                       (LENGTH FORKS)
                       (LENGTH (HAVING (FUNCTION (LAMBDA (K) (NOT (EQ R (CADDDR (CDDDR (CADR (GETQ K TYPE))))))))
                                    ARROWS))
                       (LENGTH (HAVING (FUNCTION (LAMBDA (K)(UNDEI K R))) TS)))
                 (QUOTIENT L 9.))
               (GO LIMPIO)))
        SUSPICIOUS
        (COND ((GREATERP L 25.) (PUTPROP R (QUOTE BACKGROUND)(QUOTE BACKGROUND))))
        (PUTPROP R (QUOTE SUSPICIOUS)(QUOTE SUSPICIOUS))
        (RETURN T)
        LIMPIO
        (COND ((GREATERP L 25.) (PUTPROP R (QUOTE BIGFACE) (QUOTE BIGFACE)) (PRINT (QU ((EV R) IS BIGFACE)))))
        (REMPROP R (QUOTE SUSPICIOUS))
        (RETURN ()))))
 EXPR)
```

Debugging device. BREAK is a function on file TRAC #11.

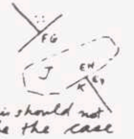
To be classified as LIMPIO, ~~and~~ a region has to have at least 1 + [1 vertex of each 9] "clean" vertices.

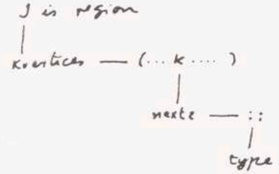
Bigfaces are limpio, too.

```
(DEFPROP GETKBA (LAMBDA (K)
  (GETQ K BACKGROUND) )
 EXPR)

(DEFPROP PUBA (LAMBDA (R)
  (COND ((NULL (FHAVING (FUNCTION GETKBA) (GETQ R NEIGHBORS))) (PUTPROP R (QUOTE BACKGROUND) (QUOTE BACKGROUND))))
 EXPR)
```

T if R has no background neighbors, marking also R as background

```
(DEFPROP BACKGROUND (LAMBDA (S)                          s = scene; (background s) gives a list of the regions belonging to the background
  (PROG (B U E REGIONS BA SO LI BACKGROUND)                                             of scene S.
        (SETQ BACKGROUND (QUOTE BACKGROUND))
        CLEAN
        (CLEAN (GETQ S REGIONS) BACKGROUND SUSPICIOUS BIGFACE BLINK)      Clean region S of previous indicators
        START
        (SETQ U (HAVING (FUNCTION SUSPICIOUS) (SETQ REGIONS (GETQ S REGIONS))))   collect in U the suspicious regions
        (CLEAN (SETQ B (HAVING (FUNCTION GETKBA) REGIONS)) SUSPICIOUS)   B are the background faces; they are no longer suspicious
        (COND (BBA1 (BREAK WITH S U B)))                     Debugging aid
        (SETQ U
              (HAVING (FUNCTION                              Keep in U those suspicious regions which are not neighbors to the
                       (LAMBDA (K)                           background; absolve the remaining of "suspiciousity"
                               (COND ((NULL(INTERSECTION B (GETQ K NEIGHBORS))) T)
                                     (T (REMPROP K (QUOTE SUSPICIOUS)) ())))))
                     U))
        (MAPC (FUNCTION
               (LAMBDA (J)
                       (MAPC (FUNCTION
stablish b-links                  (LAMBDA (K)
(background links) of                     (COND ((AND (SETQ :: (GETQ K NEXTE))
suspicious people,                                    (NOT (EQ (PROG2 (SUME K EH )(EH 7.)) J)))   <-
through T's.                                          (SUME :: EG ))
                                                (COND ((EQ J(EH 5)) (BLINK J (EG 6.))) (T (BLINK (EH 6.) (EG 5))))))))
                               (GETQ J KVERTICES))))
              U)
        (COND (BBA2 (BREAK WITH U)))                 Debugging aid
        (DO J 1 J E                                  Stop when E is TRUE   J is irrelevant.
            (SETQ E T)                               Set E to stop (unless reset in inner loop)
            (DO K
                U                                    Work on U (suspicious)
                (CDR K)                              (car K) is our current region under consideration.
                (NULL K)
                (COND ((GET (CAR K) BACKGROUND) (GO SIGUE)))
                (SETQ BA (HAVING (FUNCTION GETKBA) (SETQ :: (GETQ (CAR K)BLINK))))    :: are all regions linked to (car K); BA are background neighbors
                (SETQ SO (HAVING (FUNCTION (LAMBDA (L)(GETL L (QUOTE (SUSPICIOUS))))) ::))     SO = suspicious neighbors
                (SETQ LI (HAVING (FUNCTION (LAMBDA (L) (NULL (GETL L (QUOTE (SUSPICIOUS BACKGROUND)))))) ::))   LI = LIMPIO neighbors
                (COND (BBA3 (BREAK WITH (CAR K) BA SO LI)))          Debugging aid
                (COND (BA (GO BACKGROUND))
                      (SO (COND (LI (COND ((LESSP (LENGTH LI)(LENGTH SO)) (GO SIGUE)) (T (GO LIMPIO)))) (T (GO SIGUE))))
                      (LI (GO LIMPIO))
                      (T (GO PERDIDO)))
                BACKGROUND                                   If (car K) does not have background neighbors, mark (car K)
                (COND ((PUBA (CAR K)) (SETQ E ())) (T (GO PERDIDO)))   as background, set E to continue; otherwise do nothing.
                LIMPIO
                (COND ((REMPROP (CAR K) (QUOTE SUSPICIOUS)) (SETQ E ())))    Clean (car K) and continue through E
                SIGUE
                PERDIDO))
        (SETQ U (HAVING (FUNCTION (LAMBDA (K) (GETQ K SUSPICIOUS))) REGIONS))   Collect survivent suspicious regions, mark them
        (MAPC (FUNCTION (LAMBDA (J) (PUBA J))) U)                                as background
        (SETQ B (HAVING (FUNCTION GETKBA) REGIONS))                              B is background.
        (COND ((NULL B) (MAPC (FUNCTION                  If no background at this point, make one of the big faces to be background
                              (LAMBDA (K)(COND ((AND (GETQ K BIGFACE) (PUBA K)) (REMPROP K (QUOTE BIGFACE))))))
                             REGIONS)
                        (SETQ B (HAVING (FUNCTION GETKBA) REGIONS))))
        (PRINT (QU (SUSPICIOUS ARE (EV U))))
        (PRINTF (QU (/ / / / . THE BACKGROUND OF (EV S) IS)))
        (RETURN (PRINT B))))
 EXPR)
```



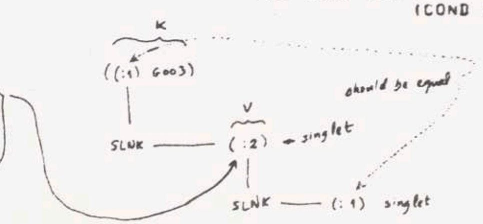


```
                                        Main function.    S: scene name.
(DEFPROP SEE (LAMBDA (S)
  (PROG (VERTICES REGIONS SLOP TYPE BODIES U LEV BACKGROUND SMB Z SWI W BASE LINEL BP SBP TRIA)
        (SETQ LINEL 120.)                                          print 120 columns
        (PRINTF (QU ((EV* PRONA) ANALYZES (EV S))))                Prona = program name
        (SETQ BASE 10.)                                            Substitute octal base by a reasonable one.
        (SETQ SLOP (QUOTE SLOP))
        (SETQ TYPE (QUOTE TYPE))
        (SETQ VERTICES (GETQ S VERTICES))
        (SETQ BACKGROUND (GETQ S BACKGROUND))
        (COND ((NULL (SETQ REGIONS (GETQ S REGIONS))) (RETURN ())))     Improper scene
        CLEAN
        (REMPROP S (QUOTE BODIES))                 Clean old properties that SEE uses internally (you are forbidden to use these names as
        (CLEAN REGIONS BODY BODY* SLNK)            indicators in property lists).
        EVIDENCE
        (PRINT (QUOTE EVIDENCE))
        (SETQ BP BPORG)
        (SETQ SBP 1)
        (MAPC (FUNCTION EVERTICES) VERTICES)                       Collect evidence at each vertex
        LOCALEVIDENCE
        (PRINT (QUOTE LOCALEVIDENCE))
        (MAPC (FUNCTION LEGS) VERTICES)                            Collect local evidence (weak links) at legs
        (PRINT (QUOTE TRIANG))
        (MAPC (FUNCTION (LAMBDA (K) (COND ((SETQ ** (TRIANG K)) (SETQ TRIA (CONS ** TRIA)))))) REGIONS)   Collect triangles
        (TRIALINK TRIA)                                            weakly link triangles that are "facing each other", etc
        (CLEAN BACKGROUND BODY)                                    Background is not part of any body
        GLOBAL                                                     AGGLUTINATION OF GLOBAL LINKS STARTS
        (PRINT (QUOTE GLOBAL))
        (SETQ R (CONS (QUOTE (())) (MAPCAR (FUNCTION (LAMBDA (J) (CONS (LIST J) (GETQ J BODY)))) REGIONS)))
        A                                           R = ( (()) . . . ( (:1) G001 G003 G027 ) . . . )  = list of nuclei
        (SETQ SWI ())                                       (:1) = regions comprised;  G001 G003 G027 = links from this nucleus (:1 in
        B                                                    this case) to other nuclei;  the whole = a nucleus.
        (COND (SWI (GO L)) ((SETQ SWI T) (PRINT R)))      SWI = T => Do not go through nuclei merging loop
        (DO (*I I)
            (R (CDR R))                                            *I is allways (cdr I).
            ((CDR *I) (CDR I))
            (NULL I)                                              <- nuclei like (() G001...) are expunged
            (COND ((NULL (CAAR I)) (RPLACD *I (SETQ I (CDR I))) (COND ((NULL I) (RETURN T))) (GO Z)))
            (DO J                                                  J moves while I stays fixed
                (CDR I)
                (CDR J)
                (NULL J)
                (COND ((NULL (CAAR J)) (GO S)) ((LESSP (LENGTH (INTERSECTION (CDAR I) (CDAR J))) 2) (GO S)))   If common links of I and J
                                                                                                                 are < 2, do nothing.
                (SETQ SWI ())                                   <- Remember to go through this loop once more  } augment 1st nucleus,
                (RPLACD (CAR J) (COMPACTIFY (APPEND (CDAR I) (CDAR J))))   <- join and compactify links       }  make it bigger.
                (RPLACA (CAR J) (APPEND(CAAR I) (CAAR J)))                 join regions                        }
                (RPLACD (CAR I) ())        } kill the other nucleus
                (RPLACA (CAR I) ())        }
                (RETURN T)
                S)                         Note that links are not destroyed, (1)--G001--(2) goes to (1 2) with a G001 loop, which is, of course, irrelevant
            Z)
        (GO B)
        L
        LOCAL                                      Taking local links into account.
        (PRINT (QUOTE LOCAL))
        (COND (Z (GO D)))                          Z is another of those switches. Things could be cleaned up here - not worth  <- ROOM FOR IMPROVEMENT
        (MAPC (FUNCTION                            LEV = ( (a b) (m n) ... ) meaning a and b, m and n, etc, are weakly linked. I was
              (LAMBDA (I)                                   I = (a b)     experimenting different ways to keep the links; this is one.
                 (AND (NOT (NULL (SETQ U                   For the strong links I use the Gensyms (see R), but there the
                          (FHAVING (FUNCTION               quantity of strong links among two regions matters, and that
                                                           is not so with the weak links. The S-links are a 3rd manner.
```

merging loop for strong links

```
                              (LAMBDA (K)           · K is a nucleus ((:1 :2) G003 G004)
(DIPI) is T if the display functions are loaded; if so,
    displays the (car I). Irrelevant for computation purposes
                                        (MEMBER (PROG2 (COND ((DIPI) (DI (CAR I)))) (CAR I))
                                                (PROG2 (COND ((DIPI)
                                                              (OR (NULL (CAR K)) (DI (CAAR K)))))
                                                       (CAR K)))))
                              R))))                    U = (((:1 :2) G003 G004)) contains the nucleus
                                                                                  owning (car I).
             (NOT (NULL (SETQ I (FHAVING (FUNCTION (LAMBDA (K) (MEMBER (CADR I) (CAR K)))) R)))) I contains the nucleus owning
                                                                                                          (cadr I)
             (NOT (EQ (SETQ U (CAR U)) (SETQ I (CAR I))))    Do nothing if both nuclei are the same.
             (GREATERP(LENGTH (INTERSECTION (CDR U) (CDR I))) 0)  Look for a strong link in addition to a weak link.
             (PRINT (QU (LOCAL ASSUMES (EV (CAR U)) (EV (CAR I)) SAME BODY)))  If found, print assumptions
             (RPLACD U (COMPACTIFY (APPEND (CDR I) (CDR U))))                      and merge.
             (RPLACA U (APPEND (CAR I) (CAR U)))
             (RPLACA I ())
             (RPLACD I ())))))
    LEV)
(SETQ Z T)
(GO A)
D
(COND (W (GO K)))                                   LIMP = T ⇒ display slowly
(COND ((DIPI) (COND (LIMP (CLEAN (GETG SCENE REGIONS) DISPL))) (DI 77) (MAPC (FUNCTION DIA) REGIONS)))
(SETQ SMB (SETQ U ()))
(QUOTE (SEPARATING R INTO SMB AND R))               Random comment.
(MAPC (FUNCTION (LAMBDA (K) (COND ((NULL (CAR K))) ((GETQ (CAAR K) BACKGROUND)) (T (SETQ U (CONS K U)))))) R)
(SETQ R U)                  Eliminate from R the background regions
(QUOTE (SINGLE BODIES WITH ONE LINK ARE LINKED))
(MAPC (FUNCTION SINGLEBODIES) R)     Merge isolated regions having exactly one strong link
(SETQ W T)
(GO A)
K
(PRINT (QUOTE SMB))            Analyzing nuclei consisting of a single region.
(SETQ SMB                      An attempt is made to join the region to some other nucleus, preferably some neighbor, as follows:
    (HAVING (FUNCTION
            (LAMBDA (J)                       (:1)                if region is alone and is not the background .....
                    (COND ((AND (SINGLET (SETQ J (CAR J))) (NULL (GETQ (SETQ J (CAR J)) BACKGROUND)))
                           (MAPC (FUNCTION                                 ..... See its neighbors — If one of them is GOODPAL
                                  (LAMBDA (K)                                                        and is not the background,
                                          (COND ((GOODPAL J K)
                                                 (COND ((DIPI) (DI J) (DIA K)))
                                                 (OR (GETG K BACKGROUND) (SLNK J K)))))))   make an S-link
                                 (GETQ J NEIGHBORS))                                        between them
                           T))))
            R))
(QUOTE (NOW SMB CONTAINS SINGLE BODIES))
(QUOTE (AHORA HAY QUE COAGULAR LA RED))                         "Now we will coagulate the network"
(COND ((DIPI) (DI 77) (MAPC (FUNCTION DIA) (GETQ SCENE BACKGROUND))))
(SETQ ::
    (HAVING (FUNCTION    ((:1) G003)
            (LAMBDA (K)
                    (AND (CAR K)
                         (NOT ((LAMBDA (V)
                                       (COND ((SINGLET V)
                                              ((LAMBDA (U)
                                                       (COND ((AND (SINGLET U) (EQ (CAR U) (CAAR K)))
                                                              (PRINT (QU (SMB ASSUMES
                                                                                  (EV* U)      If O——O, join. We know
                                                                                  (EV* V)      already they are GOODPALS.
                                                                                  SAME
                                                                                  BODY)))      Print what you did.
                                                              (SETQ ::
                                                                    (LTHEFIRST      :: is the nucleus containing (car V); it may
                                                                                    have several regions.
```

This could be part of a bigger nucleus; all we are asserting here is that it is GOODPAL of exactly one region, namely (car K)



```
                                                  K
                                                  (FUNCTION
                                                   (LAMBDA (J) (MEMQ (CAR V) (CAR J))))))
                                              (NCONC (CAR K) (CAR $$))   Merge both, by increasing (car K)
                                              (RPLACD K
                                                  (COMPACTIFY (APPEND (CDR K) (CDR $$))))
                                              (RPLACA $$ ())   } Kill ::. This also affects R, because
                                              (RPLACD $$ ())   }   :: belongs to R.
                                              ))))
                                      (GETQ (CAR V) SLNK)))))
                              (GETQ (CAAR K) SLNK))))))
                    SMB))
       RESULTS                            we are almost done ;   ;; contains nuclei which were incremented by lonely regions
       (PRINT (QUOTE RESULTS))
       (SETQ U 0)                         Fobody forms a 'body' from each nucleus.
       (COND ($$ (PUTPROP S
                        (NCONC (MAPCAR (FUNCTION (LAMBDA (W)(FOBODY (CAR W)))) $$) (GETQ S BODIES))
                        (QUOTE BODIES))
                   (PRINT (QU (THE FIRST (EV U) BODIES ARE (EV* $$)))))))
       (MAPC (FUNCTION
              (LAMBDA (J)
                    (OR (NULL (CAR J))
                        (SINGLET (CAR J))
                        (GETQ (CAAR J) BACKGROUND)                      The non-single-region bodies are
                        (AND (CPROP S (FOBODY (CAR J)) (QUOTE BODIES)) ())     also announced
                        (PRINT (QU (BODY (EV U) IS (EV* (CAR J))))))))
              R)))
 EXPR)

(DEFPROP FOBODY (LAMBDA (B)
  (LIST (SETQ U (ADD1 U)) (QUOTE BREGIONS) B) )
 EXPR)

(DEFPROP DIPI (LAMBDA ()
  (GETL (QUOTE DIA) (QUOTE (EXPR FEXPR SUBR FSUBR))) )       T if the display functions are in core
 EXPR)

(SETQ LIMP T)                                                 T for slow display
```


MIT PROJECT MAC
ARTIFICIAL INTELLIGENCE GROUP

All the evidence brought by vertex v is generated.

v = vertex

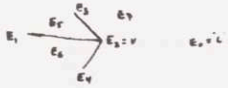
```
(DEFPROP EVERTICES (LAMBDA (V)
  ((LAMBDA (TYPE NREGIONS BACKGROUND)
        (COND ((EQ (CAR TYPE) (QUOTE FORK)) (SETQ TYPE (GETQ V KIND))
                                      (COND ((OR (KONFORK (CAR TYPE) (CADDDR TYPE))
                                                 (KONFORK (CADDR TYPE) (CAR (LAST TYPE)))
                                                 (KONFORK (CAR (CDDDDR TYPE)) (CADR TYPE)))
                                             T)
                                            ((NOBA (CAR NREGIONS) (CADR NREGIONS)(CADDR NREGIONS)))
                                            ((LAMBDA (K3 K4 K5 K6)
                                                    (AND (OR (NOSABO V V (CADR TYPE)) (GEV (CAR TYPE) K3))
                                                         (OR (NOSABO V V K4) (GEV K3 K5))
                                                         (OR (NOSABO V V K6) (GEV (CAR TYPE) K5))))
                                             (CAR (SETQ $$ (CDDR TYPE)))
                                             (CAR (SETQ $$ (CDR $$)))
                                             (CAR (SETQ $$ (CDR $$)))
                                             (CADR $$)))))
              ((EQ (CAR TYPE) (QUOTE ARROW)) (COND ((AND (EQ (PROG2 (SUME (EH 1) EF) (EF 0)) (QUOTE L))
                                                   (FORFIRST
                                                    (SETQ (EF 1) FOOP)
                                                    (FUNCTION CONTAINSV)
                                                    (FUNCTION CUENTA))
                                                   (NULL FORKS)
                                                   (NULL (CDR (HAVING (FUNCTION
                                                                        (LAMBDA (K) (PROPERARROW (EF 1) K)))
                                                                       ARROWS))))
                                              (GEV (EH 5) (EH 6)))
                                             ((NOSABO V V (EH 1)) ())
                                             (T (GEV (EH 5) (EH 6)))))
              ((EQ (CAR TYPE) (QUOTE X)) (COND ((COLINEAL
                                                 (CAR (SETQ $$ (REMOVE (EH 1) (REMOVE (EH 4) (GETQ V NVERTICES)))))
                                                 V
                                                 (CADR $$)))
                                               (T (OR (NOSABO V V (EH 1)) (GEV (EH 2) (EH 3)))
                                                  (OR (NOSABO V V (EH 4)) (GEV (EH 5) (EH 6))))))
              ((EQ (CAR TYPE) (QUOTE PEAK))(SETQ TYPE (EH 2)) (PELIN (CAR (LAST NREGIONS)) NREGIONS))
              ((EQ (CAR TYPE) (QUOTE T)) (KONFORK (EH 7) (EH 1)))
                                    (COND ((AND (NOBA (EH 5) (EH 6))
                                                (SUME (EH 1) EF)
                                                (OR (NOT (EQ (EF 0) (QUOTE T)))
                                                    (NOT (EQ (EF 1) (EH 2)))
                                                    (LEZP (EH 1) (EH 2)))
                                                (PARL (EH 5) (EH 6.) (EH 1) V))
                                           (GEV (EH 5) (EH 6))))
                                    (TESTUDY (EH 6.) (EH 4))
                                    (TESTUDY (EH 5) (EH 3))
                                    (COND ((AND (SETQ $$ (GETQ V NEXTE))
                                                (NOT (GET (EH 7.) BACKGROUND))
                                                (LEZP V $$))
                                           (SUME $$ EF)
                                           (AND (EQ (GET (EF 5) BACKGROUND) (GET (EH 6) BACKGROUND))
                                                (GEV (EF 5) (EH 6)))
                                           (AND (EQ (GET (EF 6) BACKGROUND) (GET (EH 5) BACKGROUND))
                                                (GEV (EF 6) (EH 5))))))))
   (GETQ V TYPE)
   (GETQ V NREGIONS)
   (PROG2 (SUME V EH)
         (QUOTE BACKGROUND)
         (COND ((DIPI) (COND (SBP (SETQ BPONG BP))) (SETQ SBP (NOT SBP)) (DI V) (REMPROP V (QUOTE DISPL))))))
 EXPR)
```

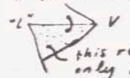
Konfork applies when, BACKGR. →

No link is put at all if one of the regions is background.

Link regions to each other. Any link may be inhibited by NOSABO

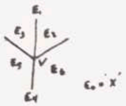
If central edge of the arrow has an L, but that L comprises only V as proper arrow, and no forks, link.

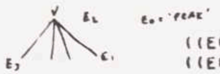




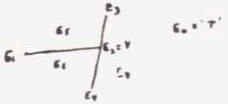
this region contains only one properarrow: V, and no forks.

Link except if forbidden by NOSABO





← Looking for BACKGR. ⟶



This LEZP avoids one of the two links in case

This LEZP avoids double links in

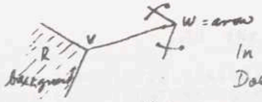
Only one of the T's makes links; the other T ducks down under LEZP

Link except background — no background

display, reset binary program space

268.

```
(DEFPROP KONFORK (LAMBDA (R W)
  (AND (GET R BACKGROUND)
       (SUME W EG)
       (EQ (EG 0) (QUOTE ARROW))
       (EQ (EG 1) V)
       (GEV (EG 5) (EG 7))
       (GEV (EG 6)(EG 7))))
 EXPR)
```

= T if 

In that case, it puts two links

Does not consult nosabo

Uses v as free variable, could be any type of vertex.

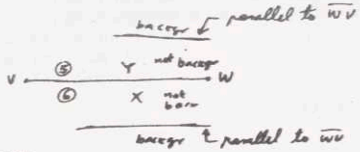
T if R is background and W is an arrow with V in the central tail. Uses EG

```
(DEFPROP NOBA (LAMBDA L (DO J L
  (SUB1 J) (EQ J 0)
        (COND ((GETQ (ARG J) BACKGROUND) (RETURN ()))))))
EXPR)
```

(noba a b c...) = T if none of a, b, c,... are background

regions

```
(DEFPROP PARL (LAMBDA (Y X W V)
  (AND (FHAVING (FUNCTION GETKBA) (GETQ Y NEIGHBORS))
       (FHAVING (FUNCTION GETKBA) (GETQ X NEIGHBORS))
       ((LAMBDA (A K L)
            (AND (PBACK (PROG2 () A (SETQ A (CAR (LAST A)))))
                 (PBACK (PROG2()(SETQ A (CTHEFIRST (GETQ X FOOP) (FUNCTION CONTAINSV))) (SETQ A (CAR (LAST A)))))))
        (CTHEFIRST (GETQ Y FOOP) (FUNCTION CONTAINSV))
        W
        V)))
 EXPR)
```

= T if 


```
(DEFPROP TESTUDY (LAMBDA (C D)
  (COND ((AND (NOBA (EM 7.) C)
              (STUDY V)
              (SUME D EF)
              (OR (NOT (EQ (EF 0) (QUOTE T)))
                  (AND (NOT (EQ (EF 1) V)) (OR (NOT (STUDY D)) (LEZP D V))))
              (PARL (EM 7.) C D V))
         (GEV (EM 7.) C))))
 EXPR)
```

Study is not working now.

```
(DEFPROP CD (LAMBDA (A) (CDAR A)) FEXPR)
```

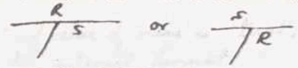
Cdr with quoted argument

```
(DEFPROP PREPARA (LAMBDA (A)          Value is 105.   1. Cleans old scene (under name "scene"; if value of 'scene' is 'trial', cleans scene 'trial')
  (PROG (B BACKGROUND W Z)                            2. Transforms a from input format to internal format
        (SETQ BACKGROUND (QUOTE BACKGROUND))
        (SETQ A
              ((LAMBDA (B) (COND ((CDR A) (REMPROP B BACKGROUND) B) (T B)))
               (COND ((ATUM (CAR A)) (CAR A)) (T (CADAR A)))))
        CLEAN
        (CLEAN (GETQ SCENE REGIONS) NEIGHBORS KVERTICES FOOP BACKGROUND SUSPICIOUS BIGFACE BLINK)
        (CLEAN (GETQ SCENE VERTICES) XCOR YCOR KIND NVERTICES NREGIONS SLOP TYPE NEXTE)
        (MAPC (FUNCTION (LAMBDA (J) (REMPROP (CADR J) (QUOTE FAS)))) TES)      Tes = ( ···· (         ) ···· )
        (SETQ TES ())                                                                         |
        (SETQ BLINK ())                                                                      Fas — (   ) → delete
        (CLEAN (LIST SCENE) REGIONS VERTICES BACKGROUND)
        (COND (BPR (BREAK IN PREPARA)))
        INITIALIZE
        (SETQ SCENE A)
        (PRINT (QUOTE LLENA))
        (LLENA (EVAL A))
        (MAPC (FUNCTION (LAMBDA (K) (PUTPROP K BACKGROUND BACKGROUND))) (GET A BACKGROUND))
        (PRINT (QUOTE FOOP))
        (MAPC (FUNCTION
               (LAMBDA (K)
                       (PROG2 (PUTPROP K
                                       (APPLY (QUOTE NCONC)                              Put foop
                                              (MAPCAR (FUNCTION CULL)
                                                      (PUTPROP K (FOOP (GETQ K KVERTICES) K) (QUOTE FOOP))))
                                       (QUOTE NEIGHBORS))
                              (PUTPROP K
                                       (APPLY (QUOTE NCONC)
                                              (MAPCAR (FUNCTION (LAMBDA (J) (CULL (CDR J)))) (GETQ K FOOP)))
                                       (QUOTE KVERTICES)))))                                            put kvertices
              (GETQ SCENE REGIONS))
        (SETQ B (GETQ SCENE VERTICES))
        (PRINT (QUOTE TYPEGENERATOR))
        (MAPC (FUNCTION TYPEGENERATOR) B)                                                Generate types
        (PRINT (QUOTE MATES))                                                            Generate classes of T's
        (MAPC (FUNCTION MATES) B)
        (SETQ Z T)
        (MAPC (FUNCTION MATES) W)
        (PRINT (QUOTE NEXTE))                                                            Generate matching T's or nexte.
        (MAPC (FUNCTION NEXTE) TES)
        (COND ((NULL (GET SCENE BACKGROUND)) (PRINTF (QU (SEARCHING FOR BACKGROUNDS OF (EV A))))    If needed, look for background
                                             (CLEAN (PUTPROP SCENE (BACKGROUND SCENE) BACKGROUND) SUSPICIOUS)))
        (RETURN 105)))
 FEXPR)

(DEFPROP UNDER (LAMBDA (R)
  (PROG2 (CUENTA (GETQ R KVERTICES)) (UNDER* TS)) )
 EXPR)

(DEFPROP UNDER* (LAMBDA (S)
  (COND ((NULL S) S) ((UNDE1 (CAR S) R) (CONS (EG 5.) (CONS (EG 6.) (UNDER* (CDR S))))) (T (UNDER* (CDR S)))))
 EXPR)

(DEFPROP UNDE1 (LAMBDA (B R)
  (AND (SUME B EG) (EQ (EG 0) (QUOTE T)) (EQ (EG 7.) R)(GOODT B) (SUME B EG)))
 EXPR)
```

```
(DEFPROP GOODPAL (LAMBDA (R S)
  (NOT (OR (MEMBER R (UNDER S))
           (MEMBER S (UNDER R))
           (AND (CUENTA (CBOUNDARY R S)) ())
           LS
           FORKS
           ARROWS
           KS
           XS
           PEAKS
           MULTIS
           (FHAVING (FUNCTION
                      (LAMBDA (K)
                              (COND ((SOME K EH) (AND (OR (EQ (EH 5) R) (EQ (EH 5) S)) (NOSABO (EH 1) (EH 1) (EH 2)))))))
                    TS))))
 EXPR)

(DEFPROP CBOUNDARY (LAMBDA (R S)
  (COND ((MEMBER S (GETQ R NEIGHBORS)) ((LAMBDA (FOOP) (CBOUND1 (CAR (LAST FOOP)) FOOP))
                                        ((LAMBDA (V) (CTHEFIRST (GETQ R FOOP) (FUNCTION CONTAINSV))) S)))))
 EXPR)

(DEFPROP CBOUND1 (LAMBDA (E L)
  (COND ((NULL L) L)
        ((EQ (CAR L) S) (CONS E (CONS (CADR L) (CBOUND1 (CADR L) (CDDR L)))))
        (T (CBOUND1 (CADR L) (CDDR L)))))
 EXPR)

(DEFPROP COMPLEMENT (LAMBDA (S U)
  (COND ((NULL S) S) ((MEMBER (CAR S) U) (COMPLEMENT (CDR S) U)) (T (CONS (CAR S) (COMPLEMENT (CDR S) U)))))
 EXPR)

(QUOTE (SPECIAL $7A68UGP $7A68UGG))

(DEFPROP FORALL (LAMBDA ($7A68UGL $7A68UGP $7A68UGG)
  (FORALL* $7A68UGL) )
 EXPR)

(DEFPROP FORALL* (LAMBDA ($7A68UGL)
  (COND ((NULL $7A68UGL) $7A68UGL)
        (($7A68UGP (CAR $7A68UGL)) (CONS ($7A68UGG (CAR $7A68UGL)) (FORALL* (CDR $7A68UGL))))))
 EXPR)

(DEFPROP FORFIRST (LAMBDA ($7A68UGL $7A68UGP $7A68UGG)
  (FORFIRST* $7A68UGL) )
 EXPR)

(DEFPROP FORFIRST* (LAMBDA ($7A68UGL)
  (COND ((NULL $7A68UGL) $7A68UGL)
        (($7A68UGP (CAR $7A68UGL)) (LIST ($7A68UGG (CAR $7A68UGL))))
        (T (FORFIRST* (CDR $7A68UGL)))))
 EXPR)

(QUOTE (UNSPECIAL $7A68UGP $7A68UGG))

(DEFPROP THEFIRST (LAMBDA (=/+$*JUL $/-$$*JL)
  (FHAVING $/-$$*JL =/+$*JUL) )
 EXPR)

(DEFPROP CTHEFIRST (LAMBDA (=/+$*JUL $/-$$JL)
```

R, S regions. True if they are good pals - Good neighbors - at the same level -

Reject if one passes under the other



The common boundary should not contain L's, forks, arrows, K's, X's, peaks, multis

Also should not contain

nosabo an impediment.

Remove 'quote' for compilation.

Remove 'quote' for compilation.

(thefirst L P) = a list with the first element of L satisfying predicate P = (fhaving P L)

```
  (COND ((SETQ =/+$*JUL (THEFIRST =/+$*JUL $/-$$JL)) (CAR =/+$*JUL)) (T (BREAK 1 (THEFIRST NULL))))
 EXPR)

(DEFPROP CONTAINSV (LAMBDA (J)
  (MEMQ V J) )
 EXPR)
```


10

END OF FILE SEE 58 F

SEECMP 57 F.

These functions are "ready to be compiled"; hence the name.

```
(SPECIAL Y* A* *D)

(DEFPROP BREAK (LAMBDA (W) (PRINT W)) FEXPR)

(DEFPROP GETQ (LAMBDA (L)
  (DISPLACE L (QU (GET (EV (CADR L)) (QUOTE (EV (CADDR L))))))
 MACRO)

(DEFPROP IF (LAMBDA (L)
  (QU (COND ((EV (CADR L)) (EV (CADDR L)))(T (EV (CDDDR L))))))
 MACRO)

(DEFPROP DO (LAMBDA (L)
  (DISPLACE L
   (PROG2 (COND ((ATOM (CADR L)) (SETQ L
                                    (CONS (CAR L)
                                          (CONS (LIST (CADR L))
                                                (CONS (LIST (CADDR L)) (CONS (LIST (CADDDR L)) (CDDDDR L))))))))
         (QU ((LAMBDA (EV (CADR L))
                    (PROG ()
                          (EV (SETQ $$ (GENSYM)))
                          (COND ((EV (CAR (CDDDDR L))) (RETURN (LIST (EV* (CADR L))))))
                          (EV* (CDR (CDDDDR L)))
                          (EV* (PROG (VARLIST STEPFN CSTEPFN SETLIST)
                                     (SETQ VARLIST (CADR L))
                                     (SETQ STEPFN (CADDDR L))
                                     A
                                     (COND ((NULL VARLIST) (RETURN SETLIST))
                                           ((NULL STEPFN) (SETQ CSTEPFN (QU (ADD1 (EV (CAR VARLIST))))) (GO B)))
                                     (SETQ CSTEPFN (CAR STEPFN))
                                     (SETQ STEPFN (CDR STEPFN))
                                     B
                                     (SETQ SETLIST
                                           (NCONC SETLIST (LIST (QU (SETQ (EV (CAR VARLIST)) (EV CSTEPFN))))))
                                     (SETQ VARLIST (CDR VARLIST))
                                     (GO A)))
                          (GO (EV $$))))
              (EV* (APPEND (CADDR L)
                          (PROG (B A)
                                (SETQ A (DIFFERENCE (LENGTH (CADR L)) (LENGTH (CADDR L))))
                                C
                                (COND ((GREATERP A 0) (SETQ B (CONS 0 B)) (SETQ A (SUB1 A)) (GO C)))
                                (RETURN B)))))))))
 MACRO)

(DEFPROP QU (LAMBDA (/+$*/-$)
  (LIST (QUOTE QU*) (CONS (QUOTE QUOTE) (CDR /+$*/-$))))
 MACRO)

(DEFPROP DISPLACE (LAMBDA (A B)
  (PROG2 (RPLACA A (CAR B)) (RPLACD A (CDR B))) )
 EXPR)

(DEFPROP QU* (LAMBDA (/-$\$$)
  (COND ((ATOM /-$\$$) /-$\$$)
        ((EQ (CAR /-$\$$) (QUOTE EV)) (EVAL (CADR /-$\$$)))
        ((AND (NOT (ATOM (CAR /-$\$$))) (EQ (CAAR /-$\$$) (QUOTE EV*)))
         (APPEND (EVAL (CADAR /-$\$$)) (QU* (CDR /-$\$$))))
        (T (CONS (QU* (CAR /-$\$$)) (QU* (CDR /-$\$$))))))
 EXPR)       (IOC 5)
```

Break is defined more extensively in File TRAC. "11 Used for diagnosis, and debugging.

Get with the second argument being quoted.

(if p q rrr) → (cond (p q) (t rrr))
↳ rrr may be a fragment.

(DO [variable or variables] [initial value(s)] [stepping or incrementing function(s)] [terminating predicate] ... body of DO ...)

If a = (1 2 3)
then (QU (a b c (ev a) d e (ev* a) F)) =
= (a b c (1 2 3) d e 1 2 3 F) "Almost" like QUOTE.

Replaces (if p q r) by (cond (p q) (t r)), allowing efficiency even when the macros are interpreted

(displace a b) = b

A, B lists
B takes the place of A. (rplaca, rplacd)
A disappears
Everybody pointing to A now points to B

```
(DEFPROP LEZP (LAMBDA (X Y) (LESSP (MAKNUM X (QUOTE FIXNUM)) (MAKNUM Y (QUOTE FIXNUM)))) EXPR)

(ARRAY EH T 20.)

(ARRAY EF T 10.)

(ARRAY EG T 10.)

(SETQ EH (QUOTE EH))

(SETQ EF (QUOTE EF))

(SETQ EG (QUOTE EG))

(DEFPROP CUENTA (LAMBDA (R)
  (PROG2 (MAPC (FUNCTION (LAMBDA (K) (SET K ()))) (QUOTE (LS FORKS ARROWS TS KS XS PEAKS MULTIS NA))) (CUENTA* R)))
 EXPR)

(DEFPROP CUENTA* (LAMBDA (R)
  (COND ((NULL R) NA)
        ((NULL (SETQ $$ (GETQ (CAR R) TYPE))) (CUENTA* (CDR R)) (SETQ NA (CONS (CAR R) NA)))
        ((SETQ $$ (GET (QUOTE (1 L LS FORK FORKS ARROW ARROWS T TS K KS X XS PEAK PEAKS MULTI MULTIS)) (CAR $$)))
         ((LAMBDA ($$) (CUENTA* (CDR R))) $$)
         (SET $$ (CONS (CAR R) (EVAL $$))))
       (T (CUENTA* (CDR R)) (SETQ NA (CONS (CAR R) NA)))))
 EXPR)

(DEFPROP MATES (LAMBDA (V)
  ((LAMBDA (TYPE)
           (COND ((AND(EQ (CAR TYPE) (PROG2 (SUME V EG) (QUOTE T)))(NOBA (EG 7.)))
                  (THROW (CADR TYPE)))))
   (GETQ V TYPE)))
 EXPR)

(DEFPROP SUME (LAMBDA (X Y*)
  ((LAMBDA (A* B)
           (COND ((STORE (Y* A*) (CAR B)) (COND ((ATOM (CADR B)) (STORE (Y* 1) (CADR B)) T)
                                                (T (MAPC (FUNCTION BRUT) (CADR B)))))))
   0
   (GETQ X TYPE)))
 EXPR)

(DEFPROP BRUT (LAMBDA (K)
  (STORE (Y* (SETQ A* (ADD1 A*))) K) )
 EXPR)

(SETQ BBA1 (SETQ BBA2 (SETQ BBA3 (SETQ SB  ())))))
```

Handwritten notes beside the listing:

- LEZP: Introduces an arbitrary but consistent lexicographical order, according to the locations of the atoms, which remain fixed in LISP 1.6.
- CUENTA: R = list of atoms, assumed vertices. Uses all of these as free variables. Its value is NA.
- CUENTA*: The value of FORKS is a list containing the vertices that are of FORK type, and similarly for the others. NA = "not accounted for" vertices – May not be vertices.
- MATES: v = vertex. If v is a T, throws v into TES = (((a b) Goo) (   ) ....) [arrow from Goo to: FAS — (m n q)]. Value is always ().
- SUME: (sume v eh) [v: vertex; eh: array name] stores in (eh 0) the type of v; in (eh 1), in (eh 2), etc., as follows: v — Type — (arrow (x y w K . .)), with arrows marking E_0 (under arrow), E_1, E_2, E_3, E_4 … (under x, y, w, K). SUME allows us to say (eh 2) instead of (cadadr (get v (quote type))).
- BRUT: This function was defined so that the compiler could compile "SUME"
- SETQ BBA1 …: Debugging variables. A break in BBA1 occurs if BBA1 = T

```
(SETQ CHENEGASU T)
```

checks negative successors (predecessors).

```
(DEFPROP LLENA (LAMBDA (A)
  (PROG (B R V)  M  (COND ((NULL A) (PUTPROP SCENE R (QUOTE REGIONS)) (PUTPROP SCENE V (QUOTE VERTICES)) (RETURN B)))
        (PUTPROP (CAR A) (CADR A)(QUOTE XCOR))  (PUTPROP (CAR A)(CADDR A)(QUOTE YCOR))
        (PUTPROP (CAR A)(CADDDR A)(QUOTE KIND))
        (SETQ B
              (CONS (LIST (CAR A)
                          (PROG2 (PUTPROP (CAR A) (CULL (CADDDR A)) (QUOTE NREGIONS))
                                 (PUTPROP (CAR A) (CULL (CDR (CADDDR A))) (QUOTE NVERTICES))))
                    B))
        (CLNK (CADDDR A))  (SETQ V (CONS (CAR A) V))
        (MAPC (FUNCTION (LAMBDA (A) (OR (MEMBER A R) (SETQ R (CONS A R))))) (CULL (CADDDR A)))  (SETQ A (CDDDDR A))
        (GO M)))
 EXPR)
```

A is a scene in input format. Transforms it into 'Internal Format'. A= (M 15.2 17.7 (:5 G :6 F :6 E) ...)

Value of LLENA is (... (M (G F E)) ....)

Puts many properties in each vertex and each region.

```
(DEFPROP CULL (LAMBDA (A)
  (COND ((NULL A) A) ((NULL (CDR A)) A) (T (CONS (CAR A)(CULL (CDDR A))))))
 EXPR)
```

Makes a list with the odd positioned elements

```
(DEFPROP CLNK (LAMBDA (B)
  (COND ((OR (NULL B)(NULL (CDR B))) B)
        ((NULL (CDDR B)) (CPROPM (CAR B) (CADR B) (QUOTE KVERTICES))
                         (CPROPM (CAR (CADDDR A)) (CADR B) (QUOTE KVERTICES))
                         (LNK (CAR (CADDDR A)) (CAR B) (QUOTE NEIGHBORS)))
        ((LNK (CAR B) (CADDR B) (QUOTE NEIGHBORS)) (CPROPM (CAR B) (CADR B) (QUOTE KVERTICES))
                                                    (CPROPM (CADDR B) (CADR B) (QUOTE KVERTICES))  (CLNK (CDDR B)))))
 EXPR)

(DEFPROP BLINK (LAMBDA (A B) (LNK A B (QUOTE BLINK))) EXPR)
```

puts a | blink — (b...) and b | blink — (a ....)

```
(DEFPROP SLNK (LAMBDA (A B) (LNK A B (QUOTE SLNK))) EXPR)
```

Lnk with indicator SLNK. (For the s. links)

```
(DEFPROP LNK (LAMBDA (A B N)
  (OR (EQ A B) (AND (CPROPM B A N) (CPROPM A B N))) )
 EXPR)
```

puts a bi-directional link between x and y, with name n. No repetitions allowed.

```
(DEFPROP LINK (LAMBDA (X Y N)
  (OR (EQ X Y) (AND (CPROP X Y N) (CPROP Y X N))))
 EXPR)
```

puts a bidirectional link between x and y, with name n. x, y, n, atoms. Allows repetitions.

```
(DEFPROP CPROP (LAMBDA (A B C)
  (PUTPROP A (CONS B (GET A C)) C)) EXPR)
```

— same as cpropm but allows repetitions

```
(DEFPROP CPROPM (LAMBDA (A B C)
(OR (MEMBER B (GET A C)) (CPROP A B C))) EXPR)
```

a | c — (b ....) ← does not put b if already there. No repetitions allowed.

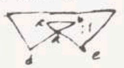
```
(DEFPROP LASFI (LAMBDA (L)
  (COND ((RPLACD (LAST L)
                 (LIST (CAR L)))
         (CDR L))))
 EXPR)
```

if L: (a b c d e)
LASFI value = (b c d e a)
modifies L to (a b c d e a)

```
(DEFPROP NEGASUCC (LAMBDA (A R)
  (COND ((GET (CONS () (GET2 A KIND)) R)) (T (BREAK 1 NEGASUCC))))
 EXPR)
```

a is vertex; R is region. Value is a vertex or ().
Value is the successor of vertex A with respect to region R in the (–) direction [↶] [having the region to the RIGHT when traveling]
Caution: answer not unique. More than one answer is possible.

(negasucc K :1) = b or d



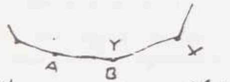

```
(DEFPROP FOOP (LAMBDA (J R)
  (COND ((NULL J) ())
        (T (CONS (SETQ ::
```

R = region
I = list of vertices belonging to R
Value = FOOP of R

```
            ((LAMBDA (X CHENEGASU)
                     ((LAMBDA (Y) (LASFI (FOOP* (CAR (NEGASUCCESSOR Y X)) Y))) (NEGASUCC X R)))
             (CAR J)
             R))
        (FOOP (COMPLEMENT (COMPACTIFY J) (CULL (CDR $$))) R)))))
 EXPR)

(DEFPROP FOOP* (LAMBDA (A B)
  (COND ((AND (EQ A Y) (EQ B X)) (NEGASUCCESSOR A B))
        (T ((LAMBDA (W) (NCONC (FOOP* (CAR W) A) W)) (NEGASUCCESSOR A B)))))
 EXPR)
```

B is the successor of A
A is the predecessor of B.

Y = initial value of B
X = successor of Y



Value is a list (starting with a vertex) which is the boundary of a simple connected curve. We will use LASFI on this value to bring it to the form of (car foop).

```
(DEFPROP NEGASUCCESSOR (LAMBDA (A B)
  (COND ((SETQ $$ (GETL (CONS () (CONS (CAR (LAST (SETQ $$ (GETQ A KIND)))) $$)) (LIST B)))
         (COND (CHENEGASU
                (COND ((EQ (CADR $$) CHENEGASU)) (T (BREAK 1 NEGASUCCESSOR $ (CHENEGASU IS NOT REGION OF A))))))
         (LIST (CADR (SETQ $$ (CDR $$))) (COND ((CDDR $$) (CADDR $$)) (T (CAR $$)))))
        (T (BREAK 2 NEGASUCCESSOR (A AND B NOT LINKED)))))
 EXPR)
```

a, b vertices. B is the negapredecessor of A or successor of A.

Value is () or a list.

Value is the successor of A in the (-) direction, with respect to the region determined by A—B and the (+) direction, plus the region between such successor and A.

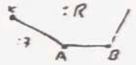
```
(DEFPROP NEXTE (LAMBDA (N)
  ((LAMBDA (FAS)
          (DO I
              FAS
              FAS
              (NULL FAS)
              (DO J
                  I
                  J
                  (NULL J)
                  ((LAMBDA (E)
                          (COND ((NULL E)(SETQ FAS (REMOVE (CAR J) FAS)) (SETQ J ()))
                                ((EQ (SETQ $$ (ESCOGE E FAS)) (CAR J))
                                 (PUTPROP (SETQ $$ (CADR $$)) (SETQ $$ (CADR E)) (QUOTE NEXTE))
                                 (PUTPROP $$ $$ (QUOTE NEXTE))
                                 (SETQ FAS (REMOVE (CAR J) (REMOVE E FAS)))
                                 (SETQ J ()))
                                ((NULL $$)(BREAK 1 NEXTE (ESCOGE E FAS) IS NULL))
                                (T (SETQ J (LIST E)))))
                   (ESCOGE (CAR J) FAS)))))
   (GETQ (CADR N) FAS)))
 EXPR)
```

N = (slope censym Goals)

NEXTE puts nexte's indicator. We will choose from (get (of that genesym) (quote fas))

1. "Y" chooses Z
2. Z chooses W
3. If W = Y, o.k. Erase both, making them matching T's, and get another new Y.
   If W ≠ Y, forget Y. Goto 1, but "Y" ← Z.

Value = (K :7)



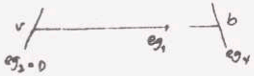
That is, standing on A, go from B to the 1st vertex in the ↺, and that is CAR of the value.
Answer is unique

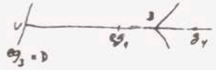
```
(DEFPROP NOSABO (LAMBDA (A *D B)
  (THROUGHTES
   A
   *D
   B
   (FUNCTION
    (LAMBDA (B)
            (OR (NOT (SUME B EG))
                (EQ (EG 0) (QUOTE L))
                (AND (EQ (EG 0) (QUOTE ARROW))(NOT (PARALLELO A B (EG 1) B)))
                (AND (EQ (EG 0)(QUOTE PEAK)) (OR (EQ (EG 1) *D) (EQ (EG 3) *D)))
                (AND (EQ (EG 0) (QUOTE T)) (NOT (EQ *D (EG 1))) (GOODT B))
                (AND (EQ (EG 0) (QUOTE K)) (OR (EQ *D (EG 3)) (EQ *D (EG 6.)))))))))
 EXPR)
```

At the beginning, *D = A
NOSABO is explained in the body of the thesis

```
(DEFPROP GOODT (LAMBDA (V)
  ((LAMBDA (D)
```

```
(COND ((NOT (EQ (EG 0) (QUOTE T))) ())
      ((GETQ (EG 7) BACKGROUND) ())
      ((GETQ V NEXTE) T)
      ((THROUGHTES
        (EG 2)
        (EG 2)
        (EG 1)
        (FUNCTION
         (LAMBDA (B)
                 (COND ((EW (PROG2 (SUME B EG)(EG 0))(QUOTE T))
                        (AND (PARALLELO V B (EG 1) B)(PARALLEL D V (EG 4) B)))
                       ((EQ (EG 0)(QUOTE X)) (OR (PARALLEL D V (EG 1) B)(PARALLEL D V (EG 4) B)))
                       (T ()))))))
       ())
      (T T)))
  (PROG2 (SUME V EG) (EG 3))))
 EXPR)
```






```
(DEFPROP PARALLELO (LAMBDA (A B C D)
  (EQ *D C) )
 EXPR)
```

*D is next to b, and between a b.

Value = { C / () }

P is predicate of one variable, a vertex
(Throughtes a d b p) returns NIL or the first vertex satisfying p.

```
(DEFPROP THROUGHTES (LAMBDA (:7AU683A *D :7AU683B :7AU683P)
  (COND ((:7AU683P :7AU683B) :7AU683B)
        ((AND (SETQ :: (GETQ :7AU683B NEXTE)) (PARALLEL :7AU683A :7AU683B :7AU683B (SETQ :: (CAADR (GETQ :: TYPE)))))
         (THROUGHTES :7AU683A :: :: :7AU683P))
        ((AND (SUME :7AU683B EF) (EQ (EF 0)(QUOTE T)) (NOT (EQ *D (EF 1))))
         (THROUGHTES :7AU683A :7AU683B (COND ((EQ *D (EF 3)) (EF 4)) (T (EF 3))) :7AU683P))
        ((EQ (EF 0)(QUOTE K)) (COND ((EQ *D (EF 6.)) (THROUGHTES :7AU683A :7AU683B (EF 3) :7AU683P))
                                    ((EQ *D (EF 3)) (THROUGHTES :7AU683A :7AU683B (EF 6.) :7AU683P))))))
 EXPR)

(DEFPROP STUDY (LAMBDA (N)
  (COND ((AND (SUME N EG )(EQ (EG 0) (QUOTE T)) (SUME  (EG 1)EG) (EQ (EG 1)(QUOTE T)) (OR (EQ (EG 3) N) (EQ (EG 4.) N)))
         (PRINT (LIST N (EG 2) (QUOTE STUDY))))))
 EXPR)
```

Study is not working. Please correct as indicated, or flush: (1) Flush (OR ...), (2) Flush (print ....) (3) Replace 1 of (eq (eg 1) ..) by
(setq :: (cond ((eq (eg 3) N) (eg 5))
((eq (eg 4.) N) (eg 6))))

Print this configuration for study.

```
(DEFPROP ESCOGE (LAMBDA (P L)
  (COND ((NULL P) ())
        (T ((LAMBDA (D E W)
                    ((LAMBDA (M) E) (MAPC (FUNCTION
                                           (LAMBDA (K)
                                                   (COND ((AND (NOT (EQ P K))
                                                               (LESSP (SETQ W (LENGH (CADR P) (CADR K))) D)
                                                               (COLINEAL (CAR P) (CADR P) (CADR K)(CAR K)))
                                                          (SETQ D W)
                                                          (SETQ E K)))))
                                          L)))
            0.1E7
            () ()))))
 EXPR)
```

P = datum
L = list of datums ("data" is the correct plural of datum)
Value is { () / Q — (datum) — where Q is the matching T of P. }

```
(SETQ TES ())
```

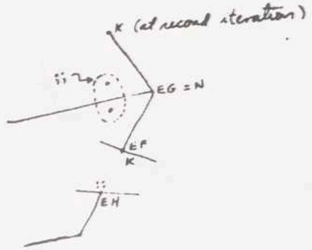
TES will hold the T's to be matching T's

```
(SETQ SCENE ())
```

Initially, scene is ().

```
(DEFPROP LEGS (LAMBDA (N)
  (AND (SUME N EG)
       (EQ (EG 0)(QUOTE ARROW))
       ((LAMBDA (Z)
               (FHAVING (FUNCTION
                         (LAMBDA (K)
                                 (THROUGHTES
                                  N
                                  N
                                  K
                                  (FUNCTION
                                   (LAMBDA (J)
                                           (AND (EQ (QUOTE L) (CAR (GETQ J TYPE)))
                                                (OR (PARALLEL N Z J (CAR (GETQ J NVERTICES)))
                                                    (PARALLEL N Z J (CADR (GETQ J NVERTICES))))))))))
                        (LIST (EG 3)(EG 4))))
        (EG 1))
       (SUME N EG)
       (LEV (EG 5) (EG 6))
       (HAVING (FUNCTION
               (LAMBDA (K)
                       (AND (SETQ $$ (GETQ K NEXTE))
                            (SUME K EF)
                            (EQ (EF 1) N)
                            (SUME $$ EH)
                            (SETQ $$ (LIST (EG 5) (EG 6.)))
                            (COND ((MEMQ (EF 5) $$) (LEV (EF 5) (EH 6.)))
                                  ((MEMQ (EF 6.) $$) (LEV (EF 6.) (EH 5)))
                                  (T (BREAK 1 LEGS))))))
               (LIST (EG 3) (EG 4)))))
 EXPR)
```

n = vertex




```
(DEFPROP HAVING (LAMBDA ($/-$$*JL =/+$*JUL)
  (COND ((NULL =/+$*JUL) =/+$*JUL)
        (($/-$$*JL (CAR =/+$*JUL)) (CONS (CAR =/+$*JUL) (HAVING $/-$$*JL (CDR =/+$*JUL))))
        (T (HAVING $/-$$*JL (CDR =/+$*JUL)))))
 EXPR)
```

(having fn L) = a list containing all elements of L (same order) that satisfy predicate FN.

```
(DEFPROP FHAVING (LAMBDA ($/-$$*JL =/+$*JUL)
  (COND ((NULL =/+$*JUL) =/+$*JUL)
        (($/-$$`JL (CAR =/+$*JUL)) (LIST (CAR =/+$*JUL)))
        (T (FHAVING $/-$$*JL (CDR =/+$*JUL)))))
 EXPR)
```

(fhaving fn L) = A list containing the first element of L that satisfies predicate FN

```
(DEFPROP ORD(LAMBDA (A B)
  (COND ((LEZP A B) (LIST A B)) (T (LIST B A))) )
 EXPR)
```

(ord a b) = (a b) if atom a < atom b
(b a) if >

Always gives a list with first element smaller than second. Establishes an arbitrary but fixed lexicographical order among atoms.

```
(DEFPROP LEV (LAMBDA (A B)
  (SETQ LEV (CONS (ORD A B) LEV)))
 EXPR)
```

local evidence

```
(DEFPROP INTERSECTION (LAMBDA (A B)
  (COND ((NULL A) A)
        ((NULL B) B)
        ((MEMBER (CAR A) B) (COND ((MEMBER (CAR A) (SETQ $$ (INTERSECTION (CDR A) B))) $$) (T (CONS (CAR A) $$))))
        (T (INTERSECTION (CDR A) B))))
 EXPR)
```

```
(DEFPROP COMPACTIFY (LAMBDA (A)
  (MAP (FUNCTION
         (LAMBDA (A)
                (COND ((OR (NULL A) (NULL (CDR A))) )
                      ((MEMBER (CAR A) (CDR A)) (RPLACA A (CADR A)) (RPLACD A (CDDR A)) ))))
        A))
 EXPR)
```

Removes repeated elements from its argument. Modifies list structure.

```
(SETQ RPAR (QUOTE /))) (SETQ LPAR (QUOTE /()))(SETQ STET (QUOTE STET))(SETQ / . (QUOTE / .))
(SETQ PREC ())
```

} Goodies for PRINTP

```
(DEFPROP PRINTF (LAMBDA (A) (PROG2 (TERPRI) A (PRINTP A)(PRINC (QUOTE / .)))) EXPR)
```

↩ Printp A ␣

```
(DEFPROP PRINTP (LAMBDA (A)
  (COND ((NULL A))
        ((ATOM A) (AND PREC (PRINC PREC)) (PRINC A) (PRINC / .) A)
        ((EQ (CAR A) STET) (PRINC (CADR A)) (PRINTP (CDDR A)))
        ((PRINTP (CAR A)) (PRINTP (CDR A)))))
 EXPR)
```

Printcs A without parentheses. No space before or after. No ↩.
The value of /␣ (normally /␣) is printed after each atom
The value of PREC (normally (), in this case nothing gets printed) is printed before each atom
If A = (... stet P Q ...) then P is printed as it is (with parentheses, etc), and without an space after it

```
(DEFPROP PBACK (LAMBDA (F)
  (COND ((NULL F) ())
        ((AND (GET (CAR F) BACKGROUND) (MPARA K L A (SLOP A (SETQ A (CADR F))))))
        (T (PBACK (PROG2 (SETQ A (CADR F)) (CDDR F))))))
 EXPR)
```

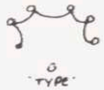
F = (:1 a :2 b ...) = car of FOOP.
Value = T if F has a segment of slope KL neighboring the background.

```
(DEFPROP SINGLEBODIES (LAMBDA (A)
  (AND (NULL (CDDR A))
       (NULL (CDAR A))
       (SETQ $$ (CADR A))
       (SETQ $$ (FHAVING (FUNCTION (LAMBDA (X) (AND (NOT (EQ X A)) (MEMBER $$ (CDR X))))) R))
       (SETQ $$ (CAR $$))
       (PRINT (QU (SINGLEBODY ASSUMES (EV (CAR $$)) (EV (CAR A)) SAME BODY)))
       (NCONC (CAR $$) (CAR A))
       (RPLACA A ())
       (RPLACD A ())))
 EXPR)
```

A is a nucleus ((:1 :2) G003 G004)
only one link
only one face or region
Look in R for another nucleus having the same link
Merge them
Kill nucleus A

```
(DEFPROP PELIN (LAMBDA (A L)
  (COND ((NULL L) T)
        ((OR (EQ A TYPE) (EQ (CAR L) TYPE)) (PELIN (CAR L) (CDR L)))
        (T (OR (NOSABO V V (GET (GETQ V KIND) A)) (GEV A (CAR L))) (PELIN (CAR L) (CDR L)))))
 EXPR)
```

L = (3 4 5 6 7) ← nregions; A → 7; ↑ type (free variable)
Links everybody cyclically except the two links to region 'TYPE'




```
(DEFPROP TRIANG (LAMBDA (R)
  (COND ((AND (EQ 3 (LENGTH (SETQ $$ (GETQ R KVERTICES))))
              (EQ 2 (LENGTH (SETQ $$ (HAVING (FUNCTION (LAMBDA (K) (EQ (QUOTE T) (CAR (GETQ K TYPE))))) $$))))
              (SETQ R (CAR (COMPLEMENT $$ $$)))
              (OR (NOT (EQ R (CADR $$))) (SETQ $$ (REVERSE $$)))
              (SUME (CAR $$) EF)
              (SUME (CADR $$) EG)
              (EQ (EF 1) R)
              (EQ (EG 1) R))
         (LIST R (CAR $$) (CADR $$) K))))
 EXPR)
```

→ R = region (TRIANG R) = (a b c R) if

and a is anything except 'T'
() otherwise.

```
(DEFPROP TRIALINK (LAMBDA (L)
  (MAP (FUNCTION
         (LAMBDA (J)
```

```
         (MAP (FUNCTION
                (LAMBDA (K)
                       (AND (PARALLEL(CAR J) (CADR J) (CADR (SETQ K (CAR K))) (CAR K))
                            (PARALLEL (CAR J) (CADDR J) (CADDR K) (CAR K))
                            (NOT (OPPOSIDE (CADDR J) (CADDR K) (CAR J) (CADR J)))
                            (NOT (OPPOSIDE (CADR J) (CADR K) (CAR J) (CADDR J)))
                            (LEV (CADDDR J) (CADDDR K)))))
              (COND (J (PROG2 () (CDR J) (SETQ J (CAR J))))))))
     L))
EXPR)
```

default value for equals — parallel tolerance — small segments are those < smase — colineal tolerance

```
(SETQ CERO 0.01)(SETQ SINTO 0.15) (SETQ SMASE 2.0)(SETQ COLTO 0.05)

(DEFPROP TYPEGENERATOR (LAMBDA (N)
  ((LAMBDA (L KIND)
           ((LAMBDA (K)
                    (PUTPROP N
                             ((LAMBDA (K1 K2 K3 K4)
                                      ((LAMBDA (K5 K6 K7 K8)
                                               (COND ((EQ (SETQ L (LENGTH KIND)) 4.)
                                                      (COND ((CROSSP N K2 N K4) (LIST (QUOTE L)(LIST K3 K1)))
                                                            (T (LIST (QUOTE L) (LIST K1 K3)))))
                                                     ((EQ L 6.)
                                                      (COND ((MPARA N (CADR KIND) N (SLOP N  (CADDDR KIND)))
                                                             (LIST (QUOTE T) (LIST K6 N(OBTU  K4 K2 K6)$$ K5 K1 K3)))
                                                            ((MPARA N (CADR KIND) N (SLOP N  (CADR K)))
                                                             (LIST (QUOTE T) (LIST K4 N(OBTU  K2 K6 K4)$$  K3 K5 K1)))
                                                            ((MPARA N (CADDDR KIND) N (SLOP N  (CADR K)))
                                                             (LIST (QUOTE T) (LIST K2 N(OBTU  K6 K4 K2)$$  K1 K3 K5)))
                                                            ((EQUAL (SETQ L
                                                                          (LIST (CROSSP K2 N K4 N)
                                                                                (CROSSP K4 N K6 N)
                                                                                (CROSSP K6 N K2 N)))
                                                                    (QUOTE (T T T)))
                                                             (LIST (QUOTE FORK) N))
                                                            ((EQUAL L (QUOTE (T T ())))
                                                             (LIST (QUOTE ARROW) (LIST K4 N K2 K6 K3 K5 K1)))
                                                            ((EQUAL L (QUOTE (T () T)))
                                                             (LIST (QUOTE ARROW) (LIST K2 N K6 K4 K1 K3 K5)))
                                                            ((EQUAL L (QUOTE (() T T)))
                                                             (LIST (QUOTE ARROW) (LIST K6 N K4 K2 K5 K1 K3)))
                                                            (T (PRINT (LIST L KIND N (QUOTE ERROR))))))
                                                     ((EQ L 8.)
                                                      (COND ((MPARA N (CADR KIND) N (SLOP N  (CADDDR KIND)))
                                                             (LIST (QUOTE K) (LIST K7 K3 K4 K5 K6 K2 K1 K8)))
                                                            ((MPARA N (CADR KIND) N (SLOP N  (CADR K)))
                                                             (LIST (QUOTE X) (LIST K2 K1 K3 K6 K5 K7)))
                                                            ((MPARA N (CADR KIND) N (SLOP N  (CADDDR K)))
                                                             (LIST (QUOTE K) (LIST K5 K1 K2 K3 K4 K8 K7 K6)))
                                                            ((MPARA N (CADDDR KIND) N (SLOP N  (CADR K)))
                                                             (LIST (QUOTE K) (LIST K1 K5 K6 K7 K8 K4 K3 K2)))
                                                            ((MPARA N (CADDDR KIND) N (SLOP N  (CADDDR K)))
                                                             (LIST (QUOTE X) (LIST K4 K3 K5 K8 K7 K1)))
                                                            ((MPARA N (CADR K) N (SLOP N  (CADDDR K)))
                                                             (LIST (QUOTE K) (LIST K3 K7 K8 K1 K2 K6 K5 K4)))
                                                            (T (PEAK N))))
                                                     (T (PEAK N))))
                                       (CAR K)
                                       (CADR K)
                                       (CADDR K)
                                       (CADDDR K)))
                              (CAR KIND)
                              (CADR KIND)
                              (CADDR KIND)
                              (CADDDR KIND))
                             (QUOTE TYPE)))
            (CDDDDR KIND)))
   ()
   (SETQ N KIND)))
 EXPR)
```

```
(DEFPROP PEAK (LAMBDA (N)
  (PROG (A C)
        (SETQ A (GETQ N KIND))
        (SETQ C (CAR (LAST A)))
        D
        (COND ((NULL A) (RETURN (LIST (QUOTE MULTI) N)))
              ((CROSSP (SETQ $$ C) N (SETQ C (CADR A)) N) (SETQ A (CDDR A)) (GO D))
              (T(RETURN (LIST (QUOTE PEAK) (LIST $$ (CAR A) (CADR A))))))))
 EXPR)

(DEFPROP PARALLEL* (LAMBDA (A B C D)
  (AND (NOT (MINUSP (DOTT A B C D)))(MPARA A B C D)) )
 EXPR)

(DEFPROP PARALLEL (LAMBDA (A B C D)
  (COND ((NULL (GETQ A SLOPES)) (PARALLEL* A B C D))
        (T (OR (PARALLEL* A B C D)
               ((LAMBDA (TOL) (AND(PARALLEL* A (SLOP A B) C (SLOP C D)) (PARALLEL* B (SLOP B A) D (SLOP D C))))
                (PLUS 0.01 (TIMES (ABS (SLOP A B)) 0.05)))))))
 EXPR)
```

} for curved segments

```
(DEFPROP MPARA (LAMBDA (A B C D)
  (EQUAL1 (SINV A B C D) 0.0 SINTO) )
 EXPR)
```

parallel without sign, only angle ≈ 0. (parallel a b c d) ≠ (parallel a b d c); (mpara a b c d) = (mpara a b d c).

```
(DEFPROP SINV (LAMBDA (A B C D)
  (QUOTIENT (CROSS A B C D) (LENGH A B) (LENGH C D)) )
 EXPR)
```

computes sin θ

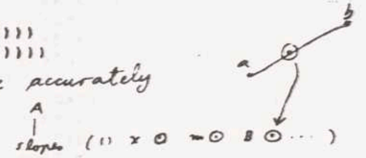

```
(DEFPROP CROSS (LAMBDA (A B C D)
  (DIFFERENCE
   (TIMES(DIFFERENCE (GETQ B XCOR) (GETQ A XCOR)) (DIFFERENCE (GETQ D YCOR) (GETQ C YCOR)))
   (TIMES(DIFFERENCE (GETQ D XCOR) (GETQ C XCOR)) (DIFFERENCE (GETQ B YCOR)(GETQ A YCOR)))))
 EXPR)
```

(cross a b c d) = cross product. a, b, c, d vertices.

dot product

```
(DEFPROP DOTT (LAMBDA (A B C D)
  (PLUS (TIMES (DIFFERENCE (GETQ D XCOR) (GETQ C XCOR))(DIFFERENCE (GETQ B XCOR)(GETQ A XCOR)))
        (TIMES (DIFFERENCE (GETQ B YCOR)(GETQ A YCOR))(DIFFERENCE (GETQ D YCOR)(GETQ C YCOR)))))
 EXPR)
```

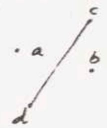
⊙ — a point such that the slope of a—b is expressed more accurately by the slope of a—⊙




```
(DEFPROP SLOP (LAMBDA (A B)
  (COND ((GET (GETQ A SLOPES) B))(B)) )
 EXPR)
```

Value = ⊙, or B, if A has no "slopes" indicator.

T if a and b are in opposite sides of c-d. a, b, c, d vertices.




```
(DEFPROP OPPOSIDE (LAMBDA (A B C D)
  (COND ((EQUAL1 (GETQ D XCOR) (GETQ C XCOR))
         (LESSP (TIMES (DIFFERENCE (GETQ A XCOR) (GETQ C XCOR))
                       (DIFFERENCE (GETQ B XCOR) (GETQ C XCOR)))
                0.0))
        (T (LESSP (TIMES (POLINEAL A C D) (POLINEAL B C D)) 0))))
 EXPR)

(DEFPROP POLINEAL (LAMBDA (B C D)
  (DIFFERENCE (DIFFERENCE (GETQ B YCOR) (GETQ C YCOR))
              (QUOTIENT (TIMES (DIFFERENCE (GETQ D YCOR) (GETQ C YCOR))
                               (DIFFERENCE (GETQ B XCOR) (GETQ C XCOR)))
                        (DIFFERENCE (GETQ D XCOR) (GETQ C XCOR)))))
 EXPR)
```

```
(DEFPROP EQUAL1 (LAMBDA L
  (LESSP (ABS (DIFFERENCE (ARG 1) (ARG 2))) (COND ((EQ L 3) (ARG 3)) (T CERO))))
 EXPR)
```

(equal1 x y ε) = T if |x-y| < ε, () otherwise. (equal1 x y) = (equal1 x y cero) ← variable

```
(DEFPROP COLINEAL (LAMBDA L (PROG
  (N A D) (SETQ D 0.0)
          (SETQ A (ARG (SETQ N 1)))
          E
          (COND ((EQ N L) (RETURN  (EQUAL1 (LENGH A (ARG N)) D COLTO))))
          (SETQ D (PLUS D (LENGH (ARG N) (ARG (SETQ N (ADD1 N))))))
          (GO E)))
EXPR)
```

(colineal a b c d e) = T if all those vertices are colineal.

```
(DEFPROP CROSSP (LAMBDA (A B C D)
  (GREATERP (PLUS CERO (CROSS A B C D)) 0.0) )
 EXPR)
```

a, b, c, d vertices. True if the cross-product is (+).

```
(DEFPROP LENGH (LAMBDA (A B)
  ((LAMBDA (##)
          (SQRT (PLUS (TIMES (SETQ ## (DIFFERENCE (GETQ A XCOR) (GETQ B XCOR))) ##)
                      (TIMES (SETQ ## (DIFFERENCE (GETQ A YCOR) (GETQ B YCOR))) ##))))
   ##))
 EXPR)
```

a —— b length of a-b. allways (+).

```
(DEFPROP REMOVE (LAMBDA (E L)
  (COND ((NULL L) L)
        ((EQUAL (CAR L) E) (CDR L))
        (T (CONS (CAR L) (REMOVE E (CDR L))))))
 EXPR)
```

(remove e l) = l without the first occurrence of element E.
E may be any lisp expression.
Uses EQUAL.

```
(DEFPROP THROW (LAMBDA (K)
  (PROG (A)
        (COND ((AND (NULL Z) (LESSP (LENGH (CAR K) (CADR K)) SMASE)) (SETQ W (CONS V W)) (GO E)))
        (SETQ A TES)
        B
        (COND ((NULL A) (SETQ TES (CONS (LIST (LIST (CAR K) (CADR K)) (SETQ ## (GENSYM))) TES))
                        (CPROP ## K (QUOTE FAS))
                        (GO E))
              ((MPARA (CAR K) (CADR K) (CAAAR A) (CADAAR A)) (CPROP (CADAR A) K (QUOTE FAS)) (GO E)))
        (SETQ A (CDR A))
        (GO B)
        E
        (RETURN ()))))
 EXPR)
```

K = datum. Value of throw is () always.
Puts K in TES

```
(DEFPROP CLEAN (LAMBDA (/-##AP68)
  (MAPC (FUNCTION (LAMBDA (K) (MAPC (FUNCTION (LAMBDA (J) (REMPROP K J)))
  (CDR /-##AP68)))) (EVAL (CAR /-##AP68))))
  FEXPR)
```

(clean b a₁ a₂ ... aₙ) b is evaluated (assumed a list)
a₁, ... aₙ not evaluated (generally atoms) indicators

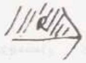
B = (b₁ b₂ ..... bₙ)
a₁ - □ ; a₂ - □ ; ... aₙ - □ (under each b) } flushes all of this

```
(DEFPROP GEV (LAMBDA (A B)
  (OR (EQ A B)
      (GETQ A BACKGROUND)
      (GETQ B BACKGROUND)
      ((LAMBDA (G) (AND (CPROP A G (QUOTE BODY)) (CPROP B G (QUOTE BODY))))
       (GENSYM))))
 EXPR)
```

puts a global evidence between a and b, except if a=b

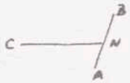
```
(SETQ BPR ())
```

```
(DEFPROP PROPERARROW (LAMBDA (R V)
  (COND ((EQ (PROG2 (SUME V EG) (EG 0)) (QUOTE ARROW))
         (NOT (EQ R (EG 7))))))
 EXPR)
```

True if  or  ; false of 

```
(DEFPROP SINGLET (LAMBDA (L) (AND L (NULL (CDR L)))) EXPR)
```

```
(DEFPROP OBTU (LAMBDA (A B C) (COND
((LESSP (DOTT A N C N) 0.0) (SETQ :: B) A)
(T (SETQ :: A) B)))EXPR)
```



Value is one of A or B, the one having an obtuse angle with NC.
The other one (the acute) is stored in ::

```
(SETQ FE (QUOTE /
```

← Form feed.

23

BIOGRAPHICAL NOTE

Adolfo Guzmán Arenas was born in Ixtaltepec, Oax., México on July 22, 1943. He entered the Escuela Superior de Ingeniería Mecánica y Eléctrica (ESIME) of the Instituto Politécnico Nacional in 1961 and, after defending the thesis "CONVERT - Design of a language for symbolic manipulation of data and of its corresponding processor", received from them the degree "Communications and Electronics Engineer" in 1965. During all his stay in the ESIME, he was receiving a scholarship from the Politécnico. He is a Registered Engineer (I. C. E.), according to Mexican law .

During 1964 (his last year in college) he held a part time programming job at the Computing Center of the Politécnico; he was sent to University of Florida (Gainesville), Stanford University (Cal.) and System Development Co. (Cal.) to learn different computing systems and languages. The first half of 1965 was spent at the Physics Department of the Centro de Investigación y Estudios Avanzados of the Politécnico, as a "technical assistant."

For his graduate studies, Adolfo Guzmán entered the Electrical Engineering Department of the Massachusetts Institute of Technology in September 1965, becoming also a member (research assistant) of the Project MAC staff, a computer-oriented inter-departmental laboratory, and became associated with the Artificial Intelligence Group of M. I. T.

After completing a thesis "Some Aspects of Pattern Recognition by Computer" he was awarded the degree of Master of Science in Electrical Engineering in 1967.

He has accepted a position as an Assistant Professor in the Department of Electrical Engineering at M.I.T. beginning February 1969. Within his research interests are computer applications and problem solving, man-machine interaction, heuristic programming and graphical information processing, the latter being the subject of his doctoral dissertation.

He is a member of the Association for Computing Machinery and the Institute of Electrical and Electronic Engineers.

Publications and technical reports

* 1. Guzmán, A. "La Estructura del Lenguaje Fortran." Presented at the I Congreso Latinoamericano sobre la Computación Electrónica en la Enseñanza Profesional. México, D.F. August 3-7, 1964.

2. Guzmán, A. Preparación de Programas para la Computadora Q-32 mediante el Sistema Telex. Program Note No. 4. Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional. Jun 1965.

3. McIntosh, H. V., Barberan, J., and Guzmán, A. LISP CONVERSION. Program Note No. 3, Centro de Invest. y Est. Avanzados del I. P. N. Feb 1965.

4. Guzmán, A. CONVERT - Diseño de un lenguaje para manipulación simbólica de datos y de su procesador correspondiente. Tesis Profesional, Escuela Superior de Ingeniería Mecánica y Eléctrica (I. P. N.) August 1965. (B.S. Thesis)

* 5. Guzmán, A. TRACE y HUSMEA: Dos funciones LISP para el rastreo de programas. Ciencias de la Información y Computación 1, 1 (Junio 1966). Universidad Nacional Autónoma de México.

6. Guzmán, A., and McIntosh, H. V. A Program Feature for CONVERT. Memorandum MAC M 305 (AI Memo 95) Project MAC,MIT. April 66.

* 7. Guzmán, A., and McIntosh, H. V. "CONVERT". Communications of the A.C.M. 9, 8 (August 1966), pp.604-615. Also available as Project MAC Memorandum MAC M 316 (AI Memo 99), June 1966.

8. Guzmán, A. Polybrick: Adventures in the Domain of Parallelepipeds. Project MAC Memorandum MAC M 308 (AI Memo 96), MIT. May 66.

9. Guzmán, A. Scene Analysis using the Concept of Model. Report AFCRL-67-0133; Computer Corporation of America, Cambridge, Mass. Jan 1967.

10. Guzmán, A. A Primitive Recognizer of Figures in a Scene. Project MAC Memorandum MAC M 342 (AI Vision Memo 119). MIT. Jan 1967.

11. Guzmán, A. Some Aspects of Pattern Recognition by Computer. M.S. Thesis, Electr. Eng. Dept. M.I.T. February 1967. Also available as a Project MAC Technical Report MAC TR 37.

12. McIntosh, H. V., and Guzmán, A. A Miscellany of CONVERT Programming. Project MAC Memorandum MAC M 346 (AI Memo 130). April 67.

* 13. Guzmán, A., and McIntosh, H. V. 'Comments on "All Paths Through a Maze" '. Proceedings of the IEEE (letters), Vol 55 No. 8 pp 1525-1527 August 1967.

* 14. Guzmán, A., and McIntosh, H. V. "Patterns and Skeletons in CONVERT". Submitted to the Second LISP Volume (book).

* Published.

15. Guzmán, A. Decomposition of a Visual Scene into Bodies. Project MAC Memorandum MAC M 357 (AI Memo 139), MIT. September 1967.

* 16. Guzmán, A. Decomposition of a visual scene into three-dimensional bodies. Proceedings of the AFIPS Fall Joint Computer Conference, Vol. 33, Part One, pp. 291-304. December 1968

* 17. Guzmán, A. Analysis of Scenes by Computer: Recognition and Identification of Objects. Proceedings of the Conference on Automatic Interpretation and Classification of Images. Pisa, Italy, Aug 26-Sept 7, 1968 (in press). [Academic Press Inc]

* 18. Guzmán, A. Object Recognition: Discovering the Parallelepipeds in a Visual Scene. Proceedings of the Second Hawaii International Conference on System Sciences. Honolulu, Hawaii, January 1969. (to appear).

---

* Published