MASSACHUSETTS INSTITUTE OF TECHNOLOGY

Artificial Intelligence

Memo No. 200

1968-1969 PROGRESS REPORT

Marvin Minsky and Seymour Papert

This report mainly summarizes the Project MAC A. I. Group work between July 1968 and June 1969 but covers some work up to February 1970. The work on computer vision is described in detail.

This summary should be read in conjunction with last year's A. I. Group Report which is included at the end of this Memo.

545 Technology Square Cambridge, Massachusetts 02139

ARTIFICIAL INTELLIGENCE GROUP

Prof. M. L. Minsky
Prof. S. A. Papert

G. K. Adler
M. D. Beeler
W. T. Beyer
T. O. Binford
Prof. M. Blum
H. E. Brammer
T. F. Callahan
E. Charniak
P. E. deCoriolis
W. Diffie
D. E. Eastlake
H. Fell
J. S. Freiberg
S. L. Geffner
J. P. Golden
R. W. Gosper
R. D. Greenblatt
J.-Y. Gresser
A. K. Griffith
Prof. A. Guzman
W. H. Henneman
A. Herskovits
C. E. Hewitt
J. T. Holloway
P. Holloway
B. K. P. Horn
K. M. Jacobs
J. L. Jaroslav
T. L. Jones
E. I. Kampits
T. F. Knight
L. J. Krakauer
Prof. W. A. Martin
G. H. H. Mitchell
Prof. J. Moses
R. Noftsker
R. Orban
D. N. Perkins, Jr.
J. S. Roe
P. R. Samson
R. C. Schroeppel
J. M. Shah
S. W. Smoliar
M. Speciner
W. A. Spies
N. F. Stone
G. J. Sussman
C. T. Waldrop
D. L. Waltz
J. C. Wentzell
J. L. White
P. H. Winston
T. A. Winograd

Guests

Prof. C. K. Chow
T. G. Evans
Prof. E. Fredkin
Prof. H. N. Mahabala
Prof. M. S. Paterson
G. Voyat

ARTIFICIAL INTELLIGENCE

This report should be read in conjunction with last year's.* This is particularly important to obtain a rounded view of our work on vision. In fact, much of what we say this year is logically prior to much that we said last year. Thus last year we discussed the abstract and theoretical problems related to the interpretation of pictures presented in a clean form (as line drawings or as subsets of an abstract retina) while this year we say more about how to obtain such pictures from the real world. The reason of course lies in the fact that the "higher level" work did not depend as heavily on prior solutions of hardware and systems problems.

We have not reported new work that lies directly in the line of development of ideas discussed last year. In particular, we have deepened and generalized some of the theorems on computational geometry. But the new state of knowledge is indistinguishable from the old on the level of discussion of this kind of report. Similarly we have not reported very new work which is still at a too primitive level of development to be presented intelligibly. This includes work on natural language processing, concept information, teaching and a number of mathematical topics. The time constant for a project of this sort is longer than a year. These matters will be dealt with in a further report due in the Fall of 1970.

Analysis of Visual Scenes: The Concept of Vertical Problem-Solving

One can use vision in many ways to find out about objects in space. Let us begin with some of the simplest ideas, and then move on to the deeper problems. If the camera can move, one could use stereoscopic correlations of two pictures. Or one might use local operations to measure motion parallax. These do not solve all problems. Stereoscopy does not work well on featureless surfaces or curved boundaries; motion effects are misleading on plain or shiny objects. Nevertheless, both methods can help find the three-dimensional locations of parts of visual objects.

Focusing provides a third method of location. Its distinctive feature is that it needs only a single eye in a fixed location, provided that the eye has a wide enough optic aperture. For then the idea of measuring distance by stereoscopic parallax merges with that of measuring distance by finding the best focus setting. Figure 1 shows two views of an object lying on a plane; in the photograph it is hard to see the object because of camouflage. (The machine sees from a little below the viewpoint of the left picture.) Berthold K. P. Horn has developed a program that can find the lens setting for best focus at each point in the visual field. The procedure, applied to a series of points along a horizontal scan through the middle of the

* See p. 43.

cube, yields the profile shown in Fig. 2. (The location of the background is less definite than that of the cube because of the background's obliquity to the camera.) Horn's program uses local Fourier transforms and compares the relative energy in the high and low spatial frequencies. It servo-controls the lens to maximize the highs, and it focuses at least as well as one can do manually.

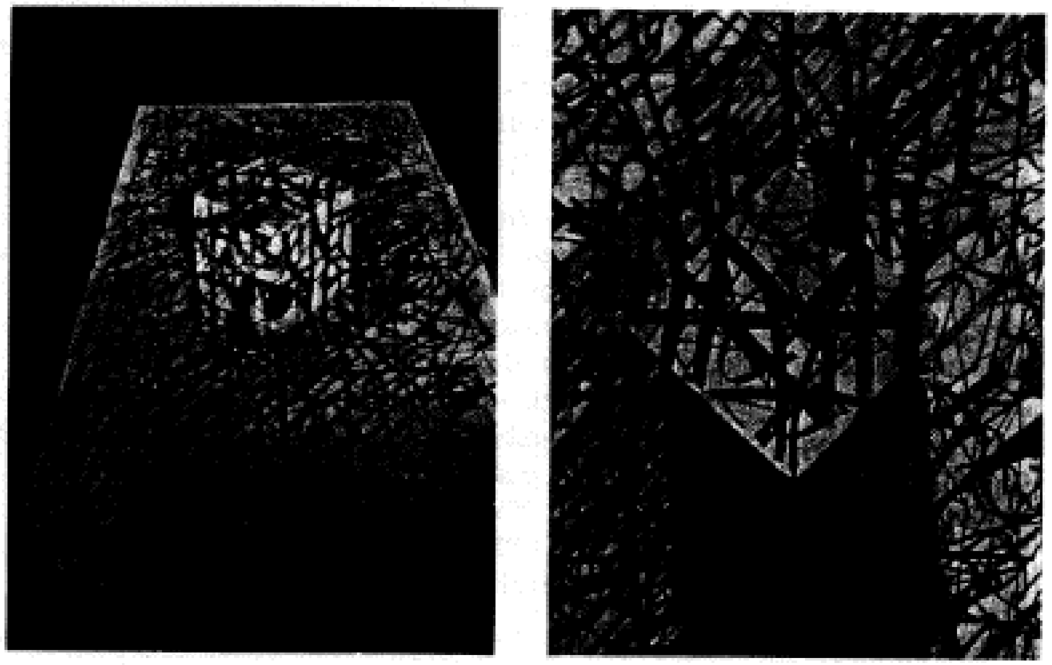


Fig. 1

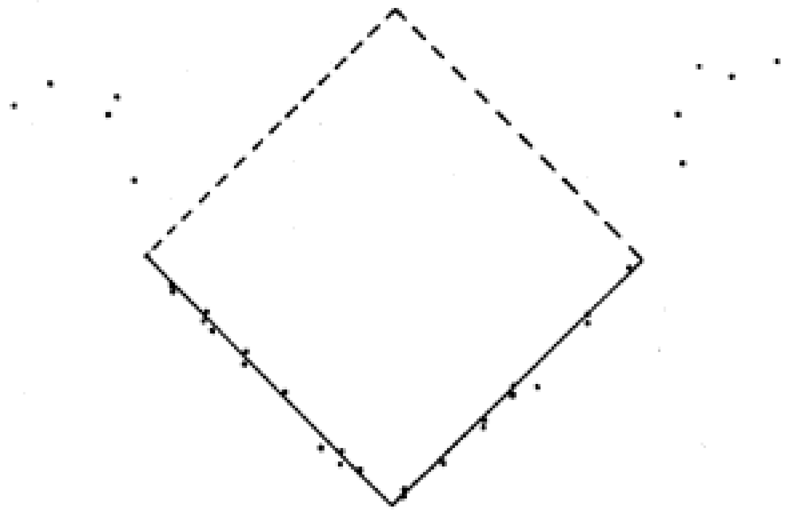


Fig. 2

Horn will discuss the focusing procedure in detail in his thesis "Shape from Shading: A Method for Finding the Shape of a Smooth, Opaque Object from One View". Two techniques are available: one uses a fast circular scan to obtain a periodic function characteristic of a large neighborhood; the other makes a faster scan of a smaller square. An operation manual (Horn, "Focusing", A.I. Memo 160) gives instructions for using the system with the PDP-6 computer and its optical-mechanical accessories, as well as some of the theory of the focusing procedure.

Now we have described three methods for optical range-finding. There are many other ways to attack just this one aspect of visual technology -- such as flying-spot scanning from a site off-center from the camera, holographic methods, and even optical radar. We must not, however, allow the myriad of technological possibilities to divert us from the deeper -- and incompletely understood -- problem of developing a visual system flexible enough to deal with real-world problems. The goal we have set ourselves is to find how to make a system that can approach the versatility of human vision. One outstanding feature of that system is its "passiveness" -- the great extent to which it can see without much special preparation or interaction with the objects in the scene. The secret lies in the intelligent viewer's ability to combine what he sees with what he <u>knows</u> about his world.

To pursue this, we have concentrated on the problem of reconstructing a three-dimensional structure using only a single monocular picture. People are quite good at understanding what is shown in a photograph; we should like to know how to make a machine do this. Now, with a very few exceptions, the many past attempts at computer analysis of scenes have been rather fruitless, and we should try to understand what went wrong. Almost all of those past attempts followed the same general plan: the picture is subjected to a sequence of transformations; each transformation is intended, in turn, to produce a successively more abstract representation until, finally, one obtains the desired description of the scene. Typically, such a sequence might be:

1) Remove noise (by clipping, smoothing, etc.);

2) Enhance features (by boosting gradients, etc.);

3) Extract features (finding edges, vertices, etc.);

4) Group features into objects (by regions, parallelisms, etc.);

5) Identify objects (by partial matches, etc.).

Although there is a great deal of plausibility to this idea of progressing relentlessly from local to global, the concept of serial stages of pre-processing does not actually work well in practice. It is simply not suited to the real problem. Errors and assumptions made at each level are passed on to the next, and even if each is quite clever at how it handles its data, the accumulation of mistakes over many stages leads to chaotic over-all results. The basic grammar of the problem is too context-dependent. The appearance of an object's features (and even their occurrence or non-occurrence) usually depends on global aspects of the arrangement and illumination of the scene. One must cope, for example, with

1) Direct line-of-sight occlusion of parts of objects,
2) Shadow occlusions that depend on the directions of lighting,
3) Highlights,
4) Reflections,
5) Textures,
6) Decorations,
7) Many other interactions between visual features and spatial forms.

Accordingly, the inevitable ambiguity problems met at each level -- "Is this an edge or not?" or "Are these two features part of the same object?" -- are often not solvable at that level. One can select a plausible interpretation only by using a wider variety of knowledge about the real world -- knowledge that ranges from principles of optics and geometry to knowledge about the particular environment and the objects likely to be in it.

In the traditional processing sequence outlined above -- we shall call it the horizontal vision system -- different kinds of knowledge are implicit at each level. The principles of optics are involved because each visual point represents a distribution function of space points in a way that depends on focus, scattering, reflection and noise. In the extraction of features, the processor must know whether the objects are likely to have straight edges, or texture boundaries, or polished surfaces. In analyzing a room, to give an extreme but real example, imagine the resulting chaos if the system did not know the significance of a picture frame!

At the level of grouping features and identifying spatial bodies, we have already seen (A.I. Group, P. R. V; page 43) how problems of projections and occlusion of three-dimensional objects lead us away from the simple template-matching schemes that work fairly well for two-dimensional problems. We found, however, that many of these difficulties could be handled by symbolic-description systems, and we shall assume that the reader is familiar with Prof. Adolfo Guzman's work, either through the survey in Progress Report V, or through his Doctoral dissertation. We now conclude that the lower-level aspects of vision, too, are best treated as problems in artificial intelligence to be handled by a mixture of general methods and special knowledge. Because the different kinds of knowledge interact at different levels, we must provide channels for such interactions so that hypotheses about high-level things like objects -- perhaps proposed by heuristics that use local evidence -- can be confirmed, rejected or revised by returning to other levels for other kinds of evidence. We use the term vertical system for this type of organization.

We have not yet enough experience with verticality to discuss it abstractly. But we now know a substantial amount about its application to visual problems; the body of this section reports what we have found so far.

Optical Anatomy of a Simple Scene

In Fig. 3 we see three cubes with dull white painted surfaces against a dark background, illuminated by concentrated light from a lamp

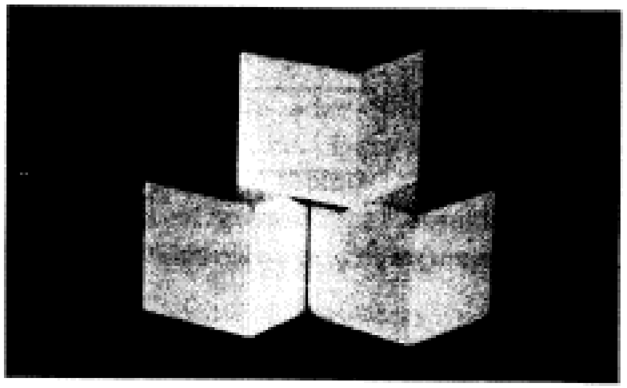


Fig. 3

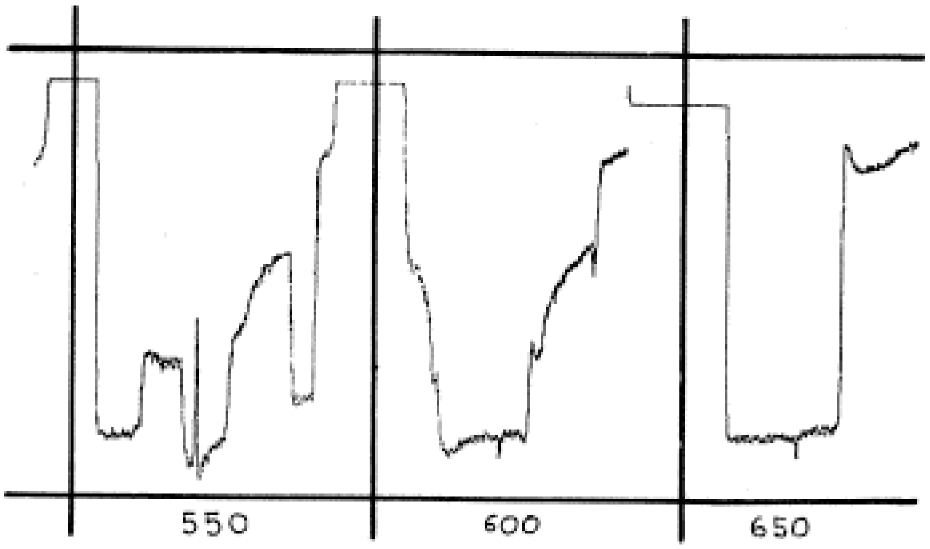


Fig. 4

above and to the right of the camera. The data and methods used here are from work by Arnold K. Griffith. Figure 4 shows three plots of the light intensity, measured along three horizontal scans, each of 1000 points across the picture.

It is difficult to discern very much in these plots. The next illustration, Fig. 5, shows the result of applying, to a sequence of such cross sections, a kind of second-derivative operation averaged over enough adjacent sample points to give flat plots over regions that have reasonably uniform gradients. (The photograph and the measurements were taken from slightly different positions.)

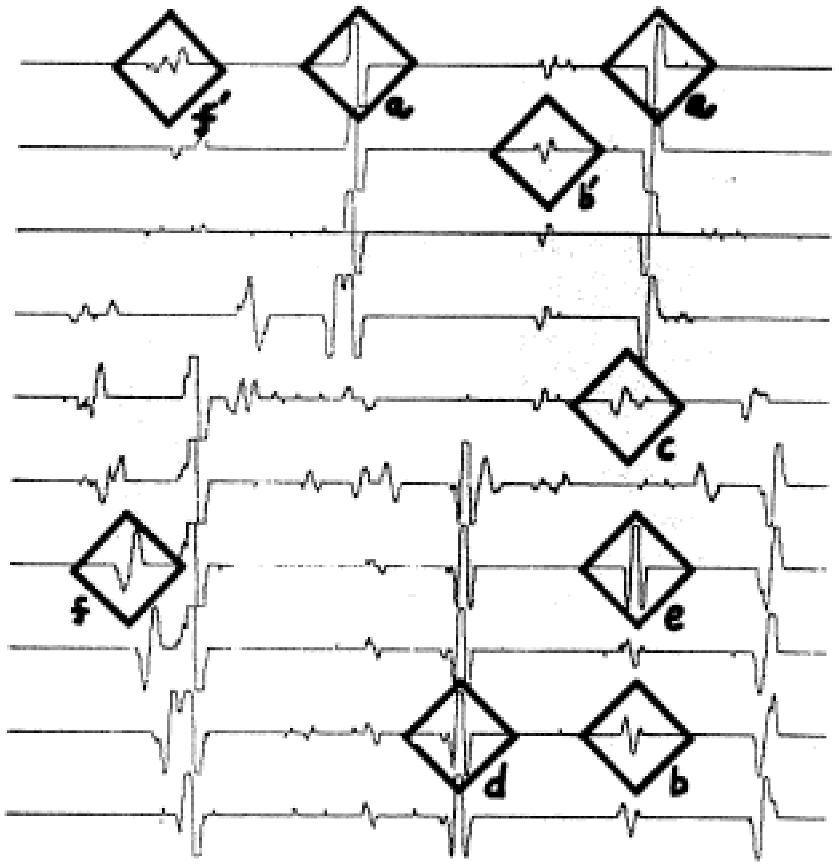


Fig. 5

We have marked a number of typical features:

> a is an outer edge. Against the dark background, it is a simple, sudden change in intensity, giving a typical "bipolar" pulse in the smoothed second derivative.
>
> b is a more symmetrical feature; it is due to the "highlight" reflection on the front edge of the lower right cube.
>
> b' is a superposition of the effects of a b and a small a.

c is the reflection of the bright surface of the upper cube in the top of the lower right cube. Although the paint there is dull, this surface is seen at a low angle, and this enhances reflections.

d is the crack between the lower cubes. The brightness measurements are all logarithmic, and truncated so that the plots won't overlap.

e is the spot of dirt on the upper front corner of the lower right cube. The edges and corners of objects often have highlights and often are dirty.

f is a shadow boundary visible on the dark background.

f' is another shadow boundary, somewhat less sharp because of penumbra and depth of focus.

The internal edges, such as at b, of uniformly colored objects often give small signals. Their width, in our plots, is an illusion due to the smoothing operation; on sharp corners, the highlight line is very narrow and easily missed by a coarse scan.

The history of attempts to write edge-finding and edge-following programs is long and inconclusive. Because the results were so obscure, Annette Herskovits (AI Memo 183) and Griffith made separate studies of the edges of geometrical objects; both concluded that the most common phenomena were superpositions of three effects (see Fig. 6).

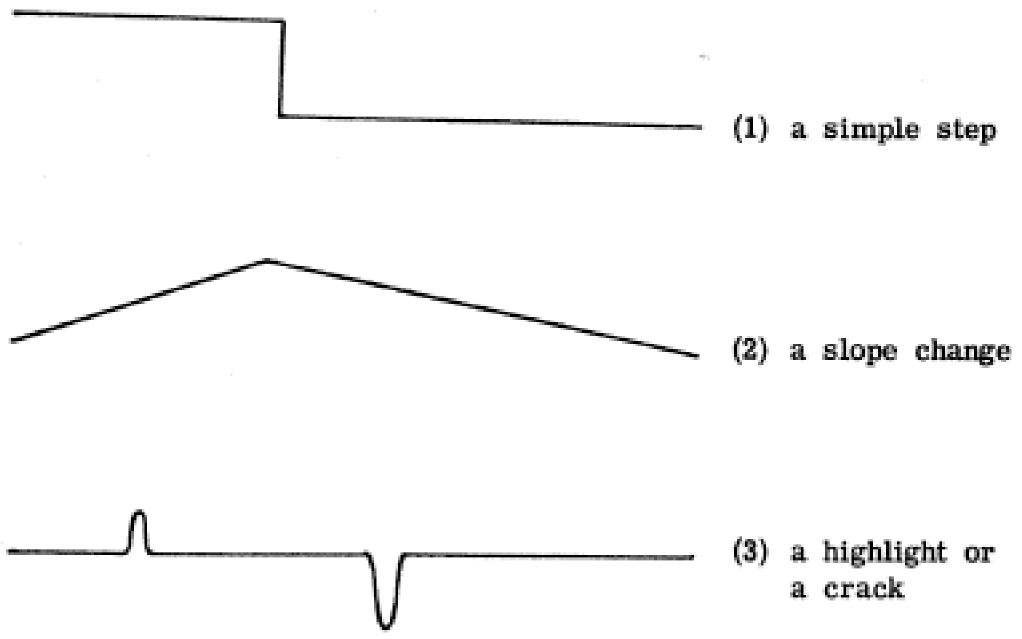


Fig. 6

They both investigated various detection filter methods for these. Griffith's thesis will include a theory of optical detection of edges under various assumptions about their intrinsic character and about the kinds of noise one might expect in a vision system. The most sensitive methods for detecting edges use two-dimensional operations, but these are very expensive with conventional hardware, because of the large amount of high-resolution information.

When a compromise must be made, it is not especially good simply to use a coarser homogeneous scan; for instead of the arrangement of Fig. 7, one can use that of Fig. 8, which has the same mean density but is better at catching thin edges.

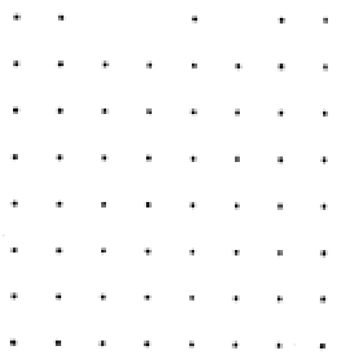


Fig. 7

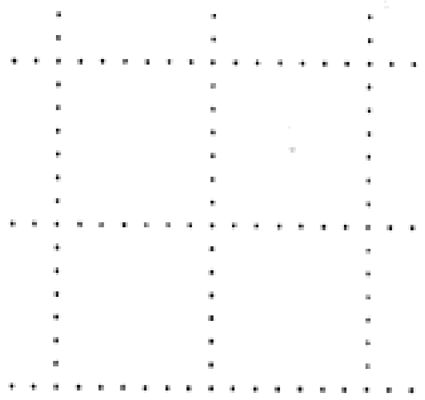


Fig. 8

Now we apply this idea to some more realistic, cluttered scenes. We mark with short dashes the local maxima of the output of the edge-filter, along a mesh of fine horizontal and vertical scans. The problem remains to convert this set of local features into lines, and then into objects. Figures 9 and 10 show the results of a process that uses a projection operation sensitive only to straight-line segments.

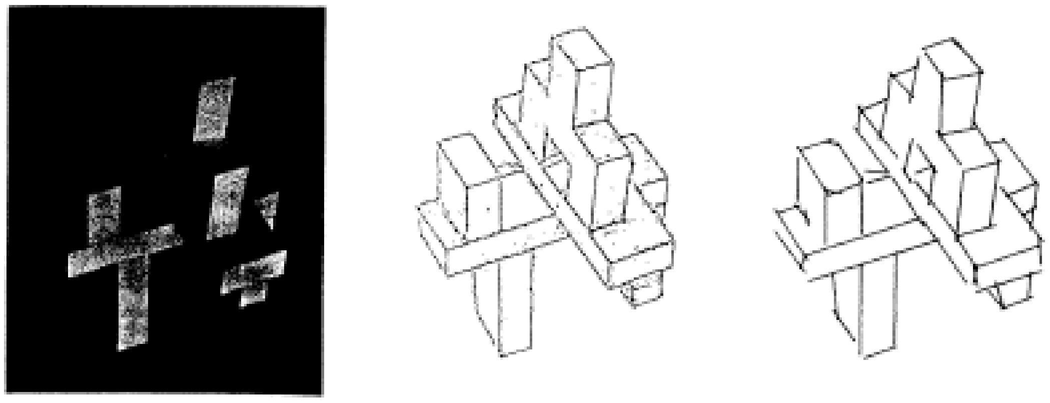


Fig. 9

Although it is not good for close-packed features, it is conservative and does not propose many false lines. Griffith's thesis will give details.

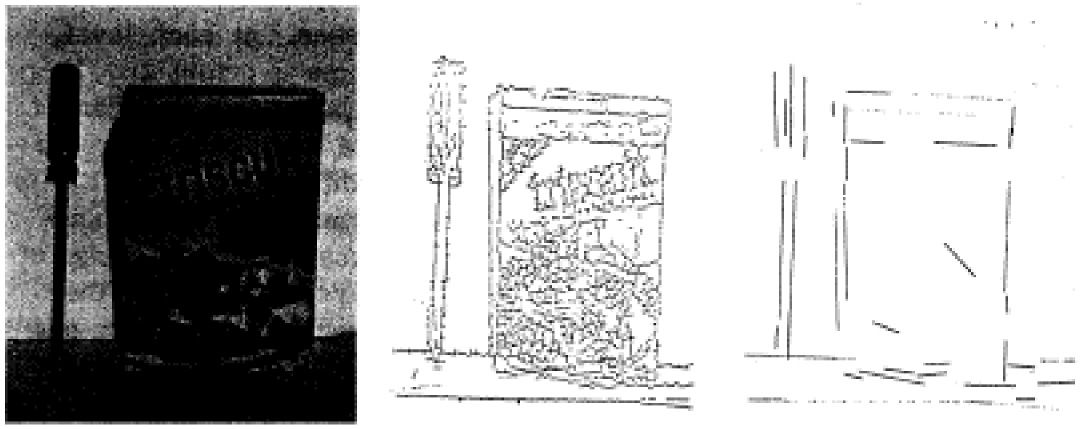


Fig. 10

Another kind of process works in a complementary manner; it labels points whose neighborhoods are relatively homogeneous. Then the system finds the boundaries of the connected regions of such points. Thomas O. Binford (AI Memo 182) describes experiments on such a system. There are many problems in deciding how to reduce the region boundaries to useful line-descriptions. We see in Fig. 12 the result of such a system applied to the relatively simple scene shown in Fig. 11.

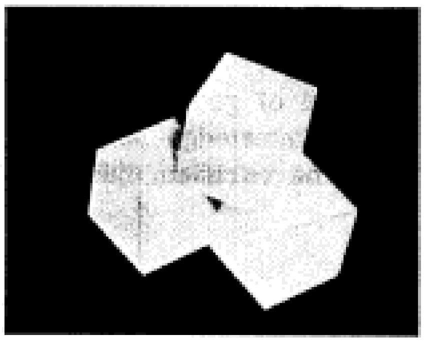


Fig. 11

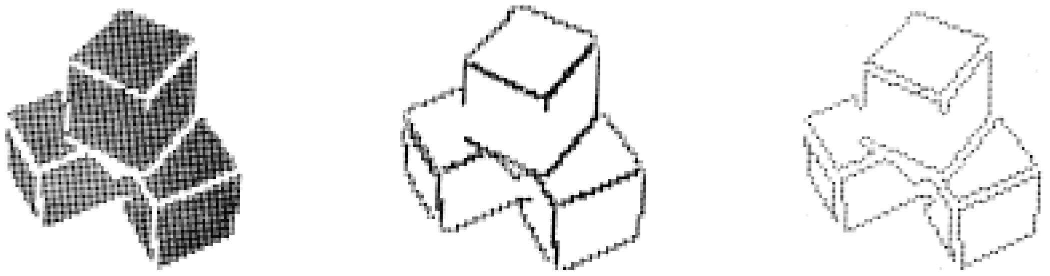


Fig. 12

Still another approach to line-finding uses a two-dimensional local-gradient detector, followed by a scheme for assigning the locally maximal features to edges, as in the early work of L. G. Roberts (Ref. 1). Richard D. Greenblatt is developing a system of this sort.

Problems exist at every stage of such processes. At each level, the selection of relevant features requires some a priori knowledge about the local world. Our verticality thesis holds that one cannot expect any one decision policy to work over a very wide range of situations, but that even a little feedback in this selection will help considerably. For example, each of the systems mentioned above will miss some edges of some objects, because of the problems of resolution, illumination, focus, contrast, texture or noise. If the system misses an interior edge, the SEE program (or rather, one version of it) may have to propose one object in place of two, as in Fig. 13, or two objects in place of one.

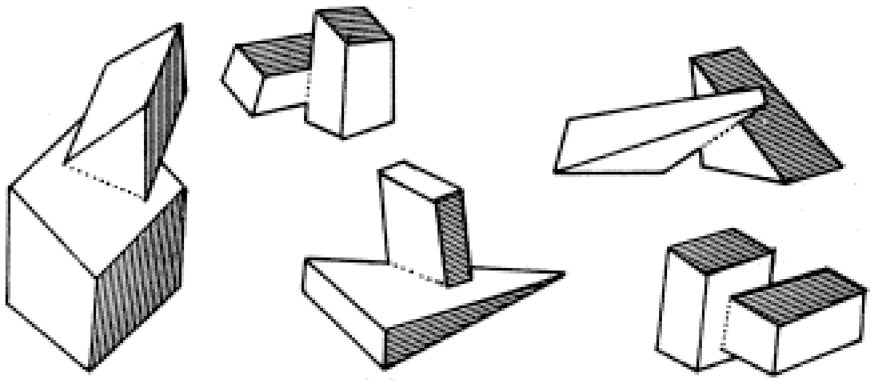


Fig. 13

Assuming we are in a world of geometrical bodies, there is a variety of ways in which to use knowledge of the fact to propose corrections. (These will have to be verified, but proposing them is most of the battle.)

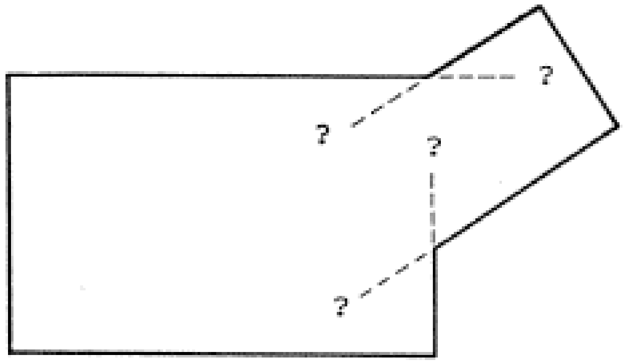


Fig. 14

Missed edges, for example, are often related to concavity (see Fig. 14), and, in a rather Bayesian way, this suggests a search for missed lines radiating from the concave vertices into the figure's interior.

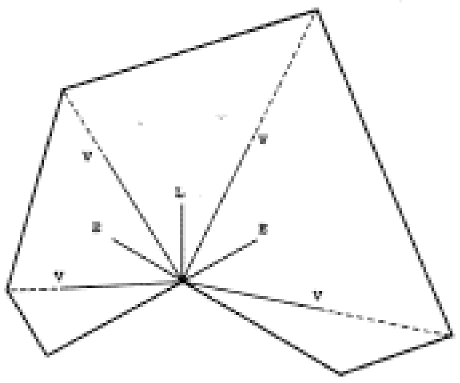


Fig. 15

In Fig. 15, we indicate proposed lines of several kinds: V directly to other vertices; E interior extensions of the vertex's edges; and L an (absolute) vertical edge (very common in real interior scenes). One might further propose parallels and lines that make confocal triplets.

It is remarkable how much can be done with such a line proposer. In the case of structures made entirely of rectangular solids, Prof. Manuel Blum showed (Fig. 16) that a remarkable number of interior edges can be reconstructed just from the outer profile of the scene.

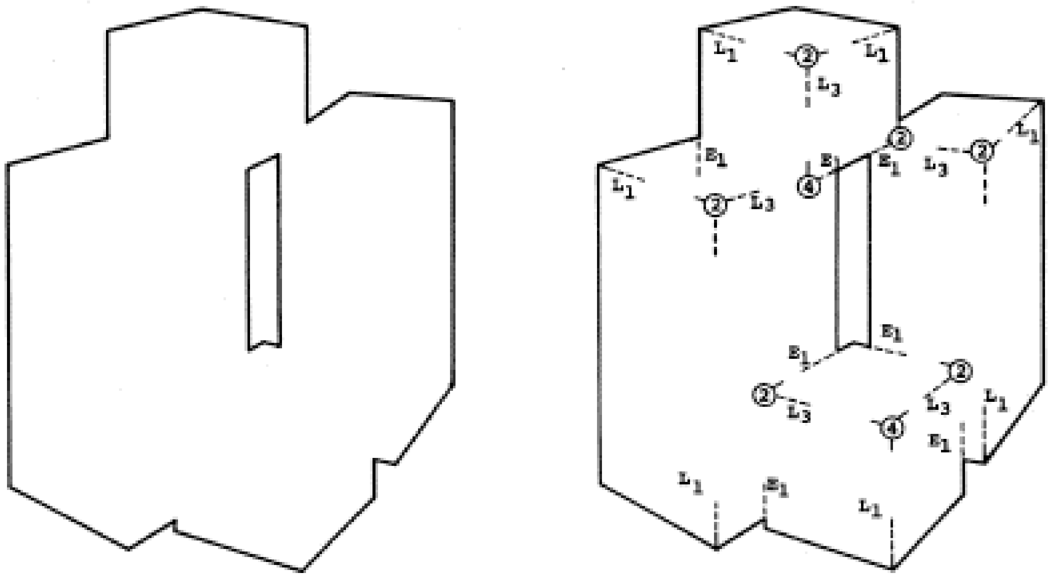


Fig. 16

The number of lines proposed by such a scheme can be held to the order of tens, rather than of thousands. And the cost of verifying the existence of an edge with a specified location is enormously smaller than that of finding all such features independently, because -- for the same statistical confidence -- rejecting a particular null-hypothesis is always much easier than screening all of a large family of possibilities. The use of proposer-verifier system can thus reduce the total picture-processing effort by relaxing the tolerances on the early, brute-force, feature-finding stages. In his forthcoming thesis, Griffith will give details of a complete system, already working, that does this. A first stage finds some of the edges in the scene. Then new lines are proposed on bases of parallelisms, region completion, etc., and verified by the detectors mentioned earlier.

By using a priori information, one can often get much more out of a picture than might seem to be in it. Assuming (correctly) that the sphere in Fig. 17(a) is uniformly colored and that the light comes from a compact source, one of Horn's programs is able to reconstruct the surface by solving the appropriate differential equations. Then this program produces the stereoscopic pair of Fig. 17(b) and 17(c). (Some readers will be able to fuse these by looking at a virtual point beyond the page.) The sharp shadow detail confirms, with the picture, the hypothesis of sharp illumination. Horn will present details of this program in his thesis. The method is quite complementary to stereoscopy since it relies on uniformity of surface, whereas stereoscopy and focusing need inhomogeneous surface detail or discontinuities.

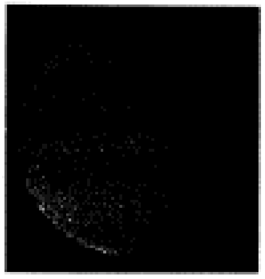


Fig. 17(a)

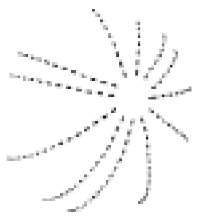


Fig. 17(b)

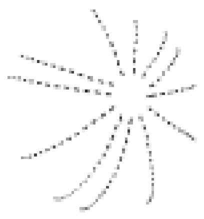


Fig. 17(c)

Even if the surface is not of uniform color and texture, there is still much more that can be done with a monocular picture. Lawrence J. Krakauer is developing a system that analyzes scenes such as a bowl of fruit. The process begins by locating and analyzing illumination maxima; it appears that, from their intensity-shape behavior one can, in many cases, distinguish dull from shiny surfaces. Local maxima connected by an illuminated band are likely to be on the same object, and Krakauer is testing some other heuristics for associating highlights with edges (Fig. 18).

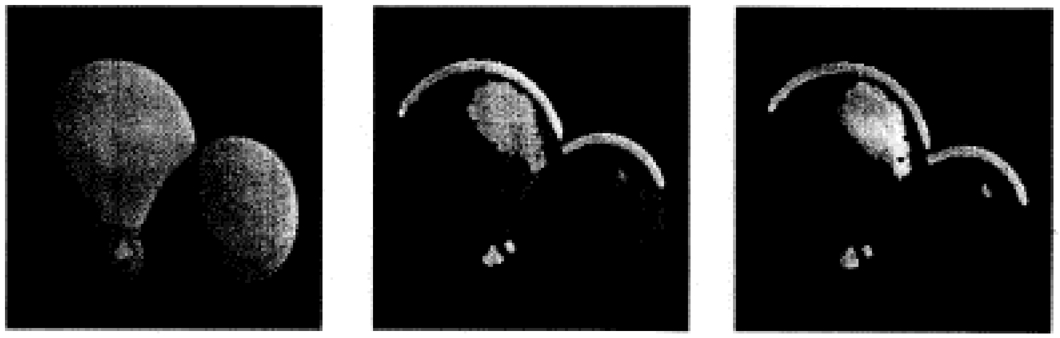

Digitized Image of Some Fruit | Edges and Maxima Enhanced | Features Alone

Fig. 18

Krakauer's goal is to discover visual characteristics of surface properties, and to establish heuristics for grouping "non-geometric" features into natural objects, perhaps as Guzman did for geometrical objects.

The Vision System

We have discussed Guzman's SEE program (see page 43); this program assumes a scene description in terms of edges, vertices and regions, and produces a proposed assignment of these features to a set of three-dimensional bodies. Guzman's thesis describes in detail a more advanced version of that project. It includes additional heuristics for linking parts of objects, analysis of the system's behavior, discussion of and heuristics for correction of mistakes because of pre-processor errors, and some analysis of the system errors due to inherent ambiguities in ordinary scenes and in a variety of standard "optical illusion" scenes. Guzman's thesis also includes some observations about the problem of matching features between stereo pairs. In particular, the following simple technique, when it is embedded in a vertical system, will solve the majority of such problems. Consider two views of a geometrical scene (Fig. 19). In any stereo pair, one can dissect the two pictures into sets of matching line-pairs, defined by the planes through the two eye-points. Any physical object visible to both eyes will be sandwiched between the same highest and lowest such lines, as suggested by Fig. 20. For two different objects, it is unlikely this will be true by coincidence, especially if the objects have more than one visible face, since the sandwich proposition is true also for each face! Thus, once enough point features are identified in the monocular pictures, one will have little difficulty in matching them between the pictures, and expensive cross-correlations should be unnecessary. This matching works well even when the two viewpoints are far apart.

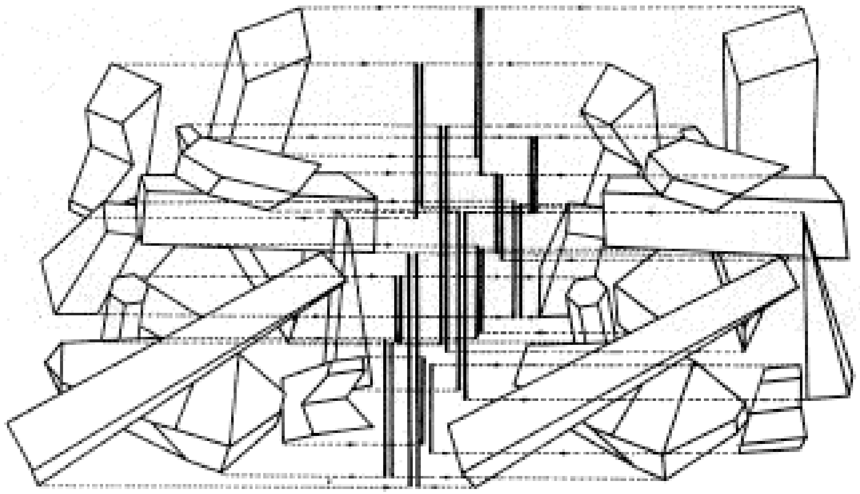


Fig. 19

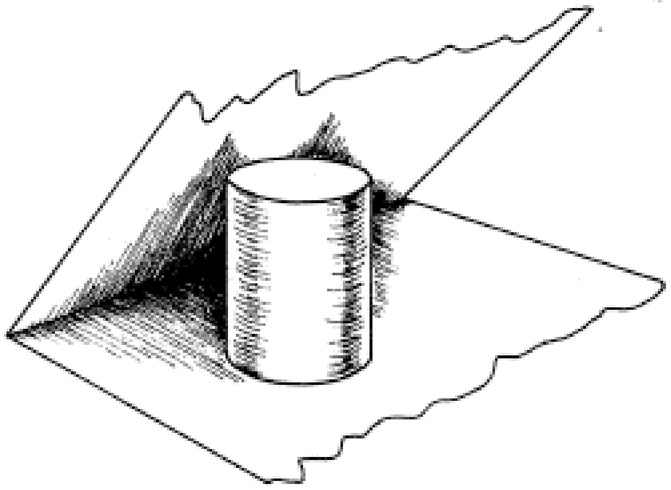


Fig. 20

In another attack on stereoscopic vision, David N. Perkins, Jr. is developing a system that compares two views and produces depth information. His matching procedure is complicated by the requirement that it be tolerant of the many kinds of errors that pre-processors are likely to make. It proceeds by matching vertices and arcs topologically, with some geometric constraints, and with a back-up and search procedure for finding maximal matches of substructures when there are possible local ambiguities. Then, given the best topological match, the program proceeds to analyze the arcs that are thus proposed for matching to obtain space curves. For example, the stereo pair of Fig. 21 is converted, by Binford's TOPOLOGIST system, to the views of Fig. 22(a) and 22(b).

Then Perkins's system, on request, produces views as seen from the right side and from above in Fig. 22(c) and 22(d) (shown here without suppression of lines that might be hidden by surfaces).

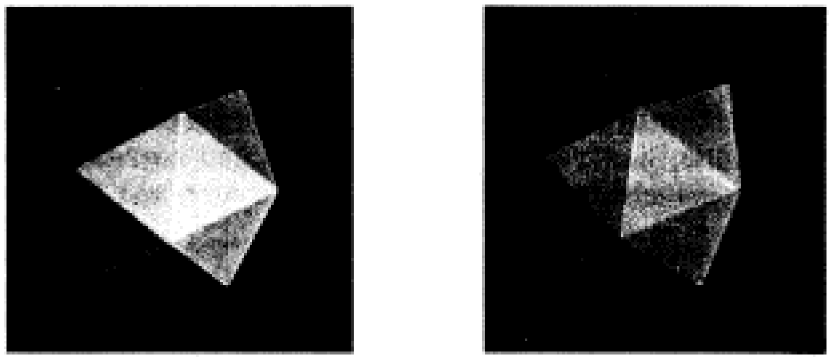


Fig. 21

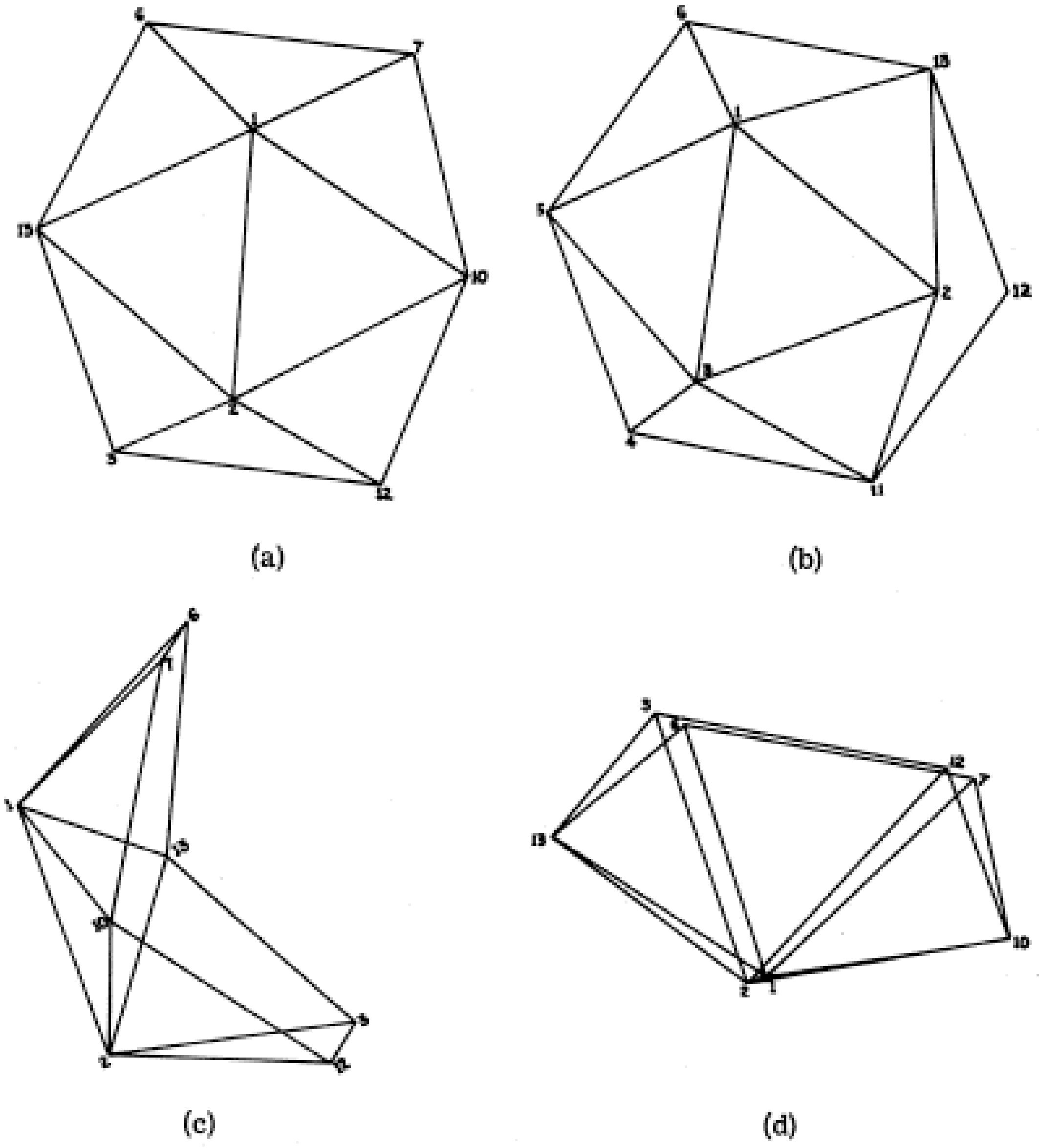


Fig. 22

We now have a complete horizontal system of programs connecting Binford's topological pre-processor with Guzman's SEE program and on through to Patrick Winston's new system (described later in the report) that recognizes some particular types of objects -- e.g., wedges, pyramids, and rectangular blocks -- and learns to identify some multi-object structure such as rows, towers and bridges.

Professor Hosakere N. V. Mahabala developed the TOPOLOGIST-to-SEE interface. This program, SETUP ("Pre-processor for Programs Which Recognize Scenes", A.I. Memo 177), gobbles a list of line segments and produces a complete topological description of the graph formed by the lines. Because such features as lines and vertices found by pre-processors have some uncertainty in location, there are usually serious problems in deciding when two edges are really the same, or in connecting edges and localizing vertices. SETUP contains heuristics for plausible guesses about such matters. All lines are treated as enclosed within strips of a certain width, and all problems about closure, intersection, containment, colinearity, etc., are resolved by procedures that are based on a single predicate about the orientation of a point with respect to one of these half-line strips. Although this may not be any better in performance than other segment-joining and line-grouping criteria, it is probably no worse, and Mahabala's system is distinctive in having greater logical clarity than any other system we have seen before. It is therefore much more likely to be improved by advances in theory!

Finding the edges themselves is still a problem. When one knows, a priori, that they are straight, the criteria Griffith and Herskovits developed are probably adequate for practical purposes, and these are further subject to substantial heuristic speed-up innovations. For the more general problem of describing an ordered set of points (such as one obtains as the boundary of a "homogeneous region") as a curve, we need a systematic way to apply various kinds of a priori knowledge. At present, we have a variety of such attempts, such as POLYSEG (Griffith, A.I. Memo 131), the Greenblatt-Holloway line-finder (A.I. Memo 101), and three others, by Binford, Greenblatt, and Jayant M. Shah. Unfortunately, none of these is well enough understood to be considered theoretically firm. A variety of other general-purpose curve-segmentation procedures has been described in the literature. But no one has really come to grips with the basic problem of incorporating the relevant a priori information, and we conclude that this is one of the aspects to be faced in designing the vertical vision system.

Richard Orban has developed a new program, ERASER, which detects and removes shadow boundaries from geometrical scenes. ERASER resembles SEE in that it uses the same classification of vertex features, but it also uses information about the relative brightness of regions. Its heuristics are based largely on the abundance of L, T and X types of vertices on the boundaries of shadow regions. ERASER is designed to criticize the output of SETUP, to remove shadow boundaries, and then to re-submit the result to SETUP before passing the problem on to SEE. It will not remove all shadows, but it is conservative about not removing real edges. Examples of ERASER's performing well are given in Fig. 23.

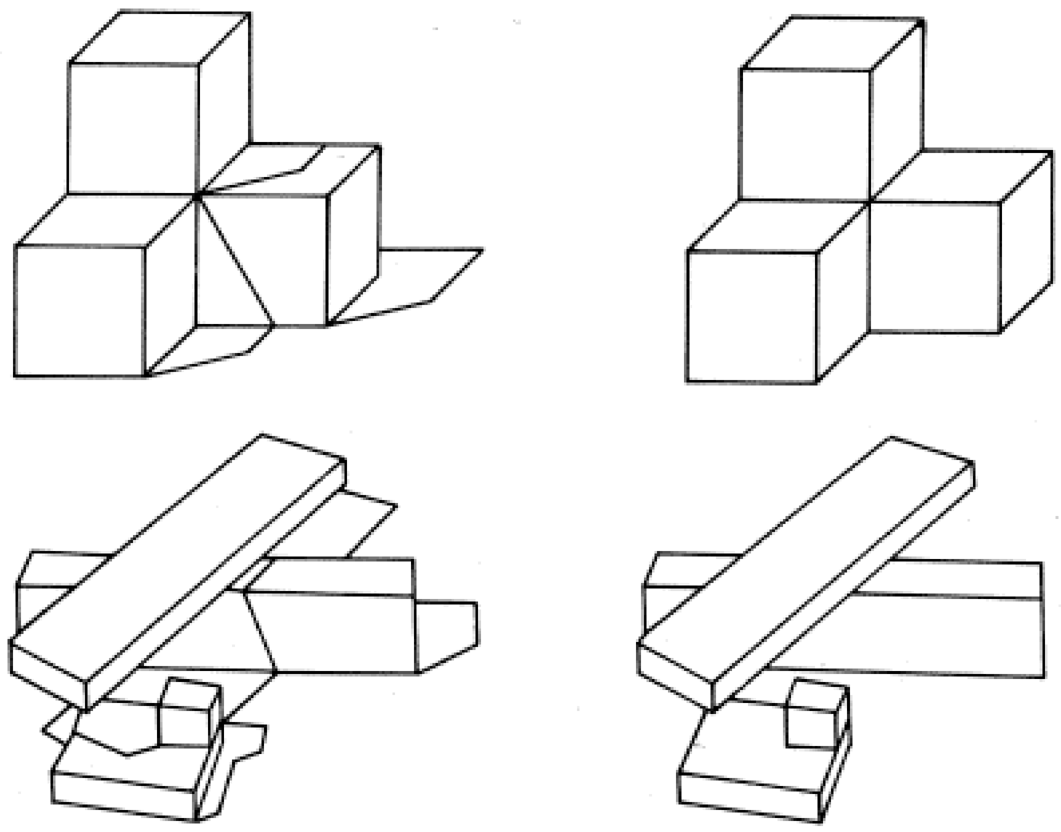


Fig. 23

The ERASER system also contains some heuristics that Binford developed for guessing the direction of illumination; this is used to bias the operation of ERASER and is available to the rest of the system. In geometrical scenes, the system can measure the angles between real vertical edges and those shadow boundaries that appear to lie on horizontal surfaces in order to get a quantitative measure of illumination angle. As an application of verticality, the operation of both ERASER and SEE could be enhanced by verifying, at a higher level, that some of the shadow boundaries lie across otherwise uniform surfaces (say, by using Perkins's stereo system) or by verifying, at a lower level, that the proposed inner shadow boundaries are less than sharp, i.e., have penumbras.

Blum and Griffith have written a program that can analyze the stability of the three-dimensional structure of rectangular blocks. The program uses this analysis in a planning scheme to propose the order in which the structure is to be built. Even in manipulating toy blocks, there are problems; one can ask which of these structures can be constructed with one hand.

Referring to Fig. 24, the one on the left cannot be built, the program asserts, because there is no stable three-block sub-part of the structure. But with another interpretation of the rules, one could first assemble the upper three blocks on the floor, then lift them into position. We expect our more advanced construction-planning programs to be able to use more advanced strategies in which subassemblies are so identified.

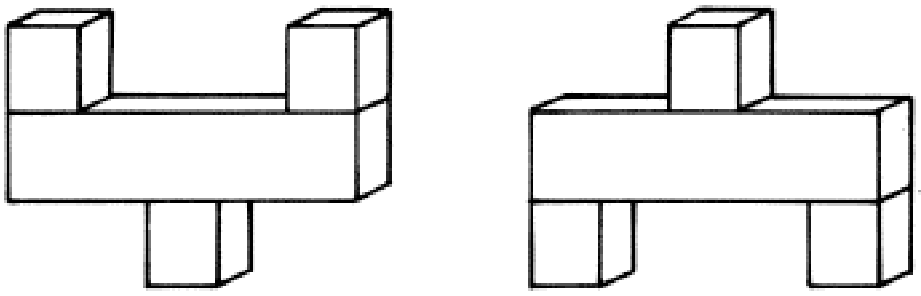


Fig. 24

Winston is completing a program that learns to recognize types of structures from sequences of examples. We consider it to be a major advance in the areas generally known as concept-formation, or machine-learning. Given a scene (represented by a collection of regions, as produced at the output of the SEE program), Winston's program attempts to describe the scene in terms of elementary objects and already-known sub-structures and relations, using a descriptive language reminiscent of that Dr. Thomas G. Evans used (Ref. 2), but Winston's language is further developed.

For example, the scene of Fig. 25 leads to a description like that shown to its right. We tell the program this is a picture of an ARCH.

Next, we inform it that the picture of Fig. 26 is not-an-ARCH. The program then proceeds to compile a new description (of ARCH), as shown to the figure's right.

This new description of ARCH is obtained by comparing the descriptions of the two scenes, thus providing a second-level description. The must-not-abut relation is the outstanding difference it discovers, and it modifies the description of ARCH to require that the abutting relation not hold between the two supporting blocks. (Winston's program is equipped ab initio with heuristics for proposing support and non-supporting contact.)

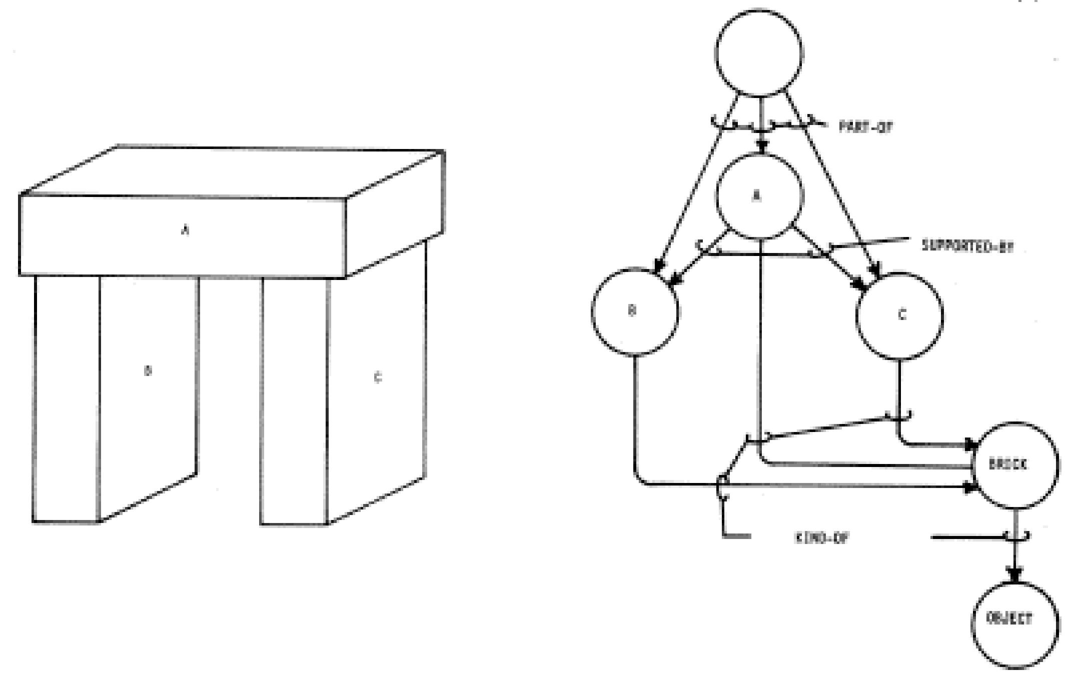


Fig. 25

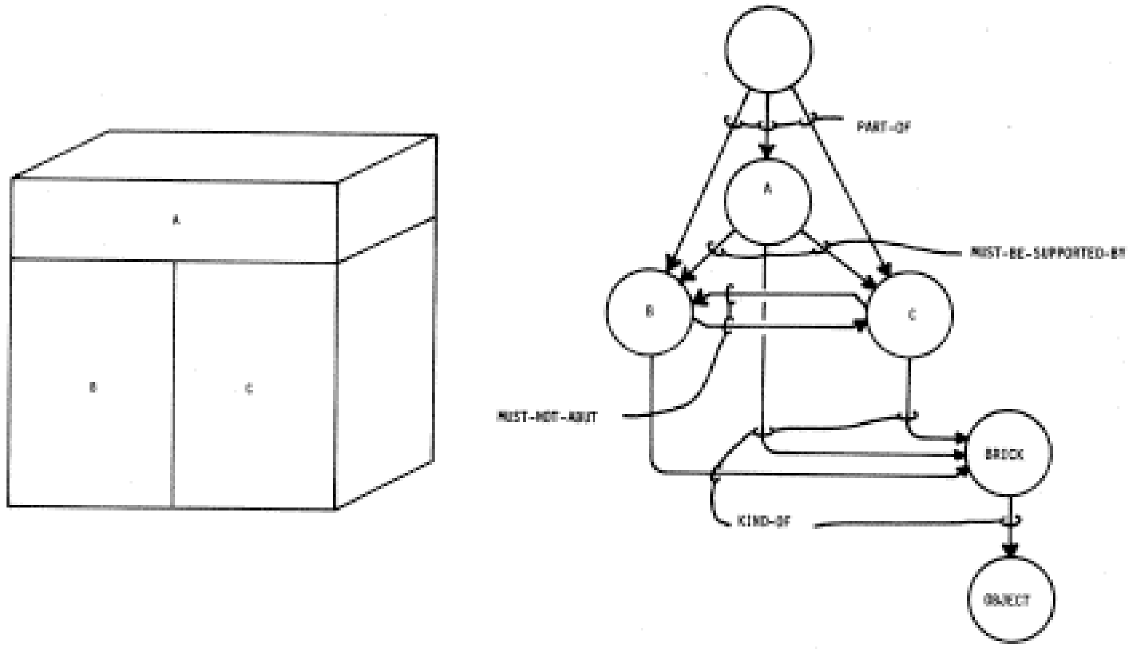


Fig. 26

Now, when we give the program the next figure (Fig. 27) as another example of not-an-ARCH, it again modifies the ARCH description, this time requiring (rather than just mentioning) the supported-by relations.

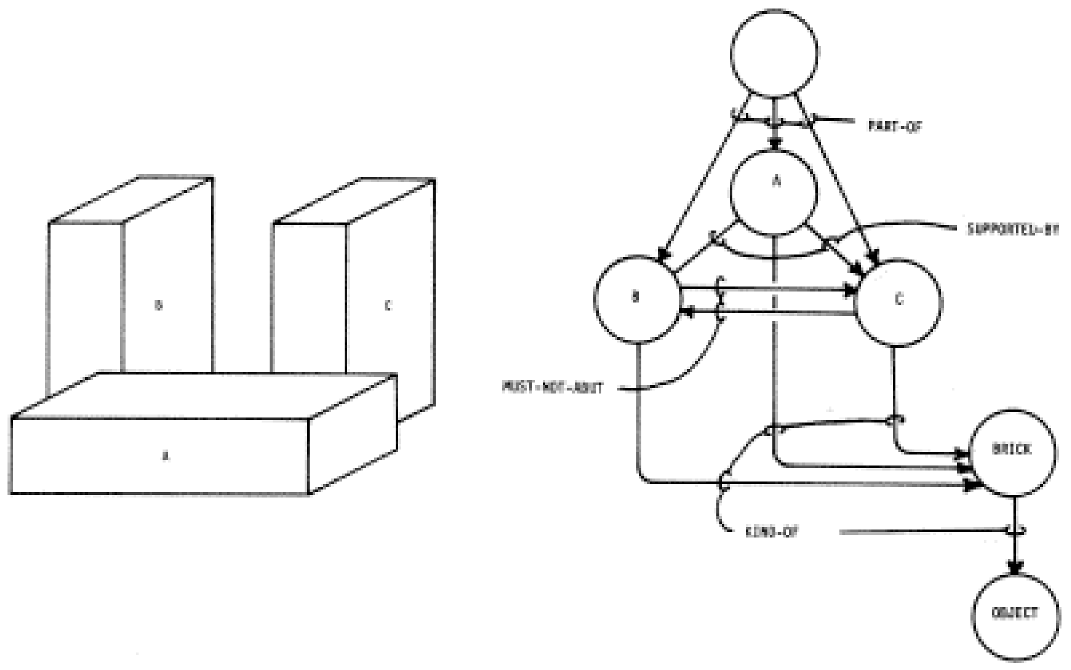


Fig. 27

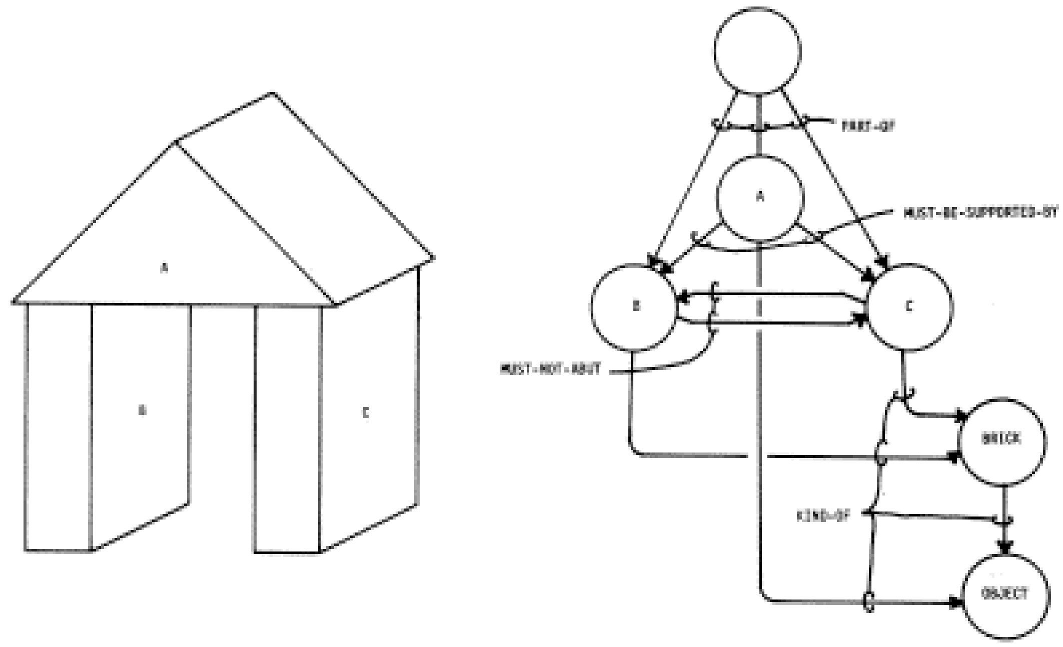


Fig. 28

Again, this change occurs because the change in support is the most prominent difference the comparison program found. Finally, we show it one more example of an ARCH, and it re-compiles a description in which the requirement that the top-object be a rectangular brick has been removed (Fig. 28).

There are many arbitrary elements in how this program decides what to do at each step -- which differences to give highest priorities, how to match up different description networks, what explanations or excuses should be assigned to the differences it notices. These commitments are kept on back-up trees, so that it is possible for the program to recover from at least some disasters. The goal is to obtain behavior that would be plausible in, if not typical of, a child. The particular concept that the program develops from a certain sequence of examples will depend very much on those examples and on the order in which they are presented -- as well as on the connection of concepts the program has already acquired at that time. The experimenter will not always get the results he wants or expects! We cannot expect the first real concept-learning programs to be foolproof, any more than a teacher can expect his favorite instructional technique always to work; but here at last we have a chance to understand precisely how a training sequence interacts with built-in and previously acquired ingredients of the system. Winston's methods are related to earlier ideas like those in Newell (Ref. 3) and emphasize the use of explicit descriptions. While there is some similarity, in strategy, to the Feigenbaum-Simon EPAM scheme (Ref. 4), the result is pointed toward a true structural network description of the concept. It should therefore lend itself better to direct analyses of examples, instead of having to work through synthesis by generating and testing proposed examples.

More generally, having a description (rather than just a test) for a concept seems absolutely crucial. The advantages include:

1) The possibility of making deductions about the concept, including its consistency with a proposed detection scheme;

2) Combining several descriptions in non-trivial ways;

3) Comparing and contrasting descriptions, as in Evans's program;

4) Using the description to generate, rather than merely to select, what next to do in a search program.

Similarly, in search processes, one could combine several descriptions to obtain summaries of the information acquired in exploring different alternatives. In most earlier heuristic search schemes, the procedures usually have to abandon almost all information acquired in the course of unsuccessful exploratory attempts because of inadequate descriptive facilities.

The A.I. Group is now committed to a broad attack on the problems of symbolic learning, through the application of heterological kinds of knowledge to the analysis of descriptions. An essay (A.I. Memo 185) gives a preliminary statement of our plans for this project.

Theorem-Proving

Gerald J. Sussman has implemented a theorem-proving program that uses the Resolution principle with an assortment of retrieval methods and other heuristics to make deductions in predicate calculus. Although there has been a number of interesting results, we nevertheless believe that the use of this technique, as an approach to artificial intelligence, is receiving much undue attention today, and we do not plan to give it a large place in our future activity unless some new and impressive demonstration of its power comes to light. Sussman has been experimenting with a variety of means for combining the deductive strength of predicate-calculus resolution with heuristic flexibility of less formally constrained problem-solving schema, but his conclusions are not encouraging. An example that puts one of the problems into a nutshell is this: suppose one is given

$A \Longrightarrow B$ and

$C \Longrightarrow D$, and

$(A \Longrightarrow B)\ \&\ (C \Longrightarrow D) \Longrightarrow E$,

and one wants to deduce the simple conclusion

E.

To be sure, the program manages eventually to obtain E, but only after many steps of converting the given statements into expanded "normal" forms that are in themselves rather meaningless.

This would not be so bad in itself, but it would become an acute problem if one were to try to give the theorem-prover additional advice about what to do in other situations, since the kinds of situations for which we can describe such advice are hard to represent in terms of the fractured internal expressions the Resolution system creates for its own use. In another experiment Sussman modified the representation of this problem so that the $\Longrightarrow$ symbols were not recognized by the prover as meaning implies, and he added a separate set of axioms for using a more natural deduction method. Now the Resolution system produced a better proof in less time, even though it had to work indirectly through the new axioms! No doubt this particular problem could be ameliorated by some variant of the many combinatorial schemes that are being widely studied today, but we feel that, once such systems attempt to solve "real" problems, all such devices will fail as mere stop-gaps; they do not help one to come to grips with the construction of the kinds of cognotive models we think are needed to solve hard problems by using accumulated knowledge and experience. Sussman's system takes some steps in this direction by maintaining its statements in a structure partially ordered by the substitution-instance relation. Nor is his system restricted to first-order predicate calculus. But our gloomy expectations remain. Incidentally, we do not feel that completeness, or even consistency, is of very large importance. Logical completeness is more or less inevitable in any system that knows a great deal, and consistency is not a notable feature in the intelligent machines that already exist. (The backers of the Resolution method achieve the wrong kind of completeness; the kind of completeness we need is for the problem-solver to be able to use any "natural" technique.)

At perhaps another extreme, Carl E. Hewitt is developing a language for constructing deductive systems with the utmost heuristic flexibility. His language, PLANNER, is designed to allow use of knowledge and types of representations of great variability. PLANNER must be one of the least procedural languages: to a large extent, one can specify what one wants done rather than how to do it. Consider, for example, a statement of the form "A implies B". As it stands, it is a simple declarative statement. But in PLANNER it can instead be interpreted as the imperative: "set up a procedure that will see if A is ever asserted, and if this happens assert B also." Or, it can be interpreted: "set up a procedure that will see if B is ever desired as a goal, and if so assert A as a new subgoal to be deduced." This is only the skeleton of the idea: PLANNER contains machinery for easy manipulation of many different roles of declaratives, imperatives, goals and deductions. In attempting a particular deduction, for example, programs can specify suggestions for other theorems that should be used (and even in what order) to make the deduction. All statements are expressed in a powerful new pattern-matching language, MATCHLESS, in which control of procedures is specified in terms of the forms of assertions, rather than in terms of particular assertions.

Because problem-solving often requires building up elaborate temporary structures, PLANNER has machinery for handling statements that were once true in a model which may no longer be true after actions have been performed, and for drawing conclusions that may have to be deduced from such a change. Control of such matters is specified in terms of local states, to which are bound information about changes in the data base -- erasures, assertions, new definitions, etc. -- since the data were last updated.

Hewitt describes his system qualitatively in a conference paper, and in detail in A.I. Memo 168. He has now implemented the language and is programming a compiler to obtain sufficient speed to permit full-scale experiments. At present, he is using it to study deductions about the manipulations of objects by a robot, as in "pick up all pairs of cubes that are the same color, with one cube on top of a third cube that is in front of the other." Terry A. Winograd is completing a system that will translate such natural English statements into PLANNER assertions and theorems.

Natural Language Systems

Winograd is completing a system for handling problems of computer understanding of natural language. The system is designed to accept information in normal English sentences, to answer questions, and to execute commands, using semantic information context to understand pronoun references and to disambiguate grammatically complicated texts. To do this, Winograd uses a new linguistic scheme, based partly on the systematic grammar described in Halliday (1967, Ref. 5) and in Winograd (1968, Ref. 6) and partly on a special representation for both the grammar and the semantics. In previous attacks on such problems, some workers have used heuristic tricks -- key words or matching of phrase fragments -- or they have used non-heuristic formal grammars to do a detailed analysis of the sentence to which the semantics are to be applied. Winograd's system is based on a heuristic grammar that uses contextual information in analyzing the sentence, carrying out the semantic analysis concurrently; this is made possible by representing the grammar as a program instead of as a set of static rules.

Definitions of words, as well as the system's knowledge about things, are also stored as programs and are available to a deductive part of the system. This allows much more flexibility than one gets naturally from networks or rule-lists. The program can make long, complex deductions in answering questions or in absorbing new information.

The grammatical part of the system is operating, with a quite comprehensive English grammar (for a description, see Winograd, "PROGRAMMAR, A Language for Writing Grammars", A.I. Memo 181). The semantic programs are still in preparation, to be combined with the deductive system, which will use Hewitt's PLANNER language. The entire system could be used for a general question-answering facility for any corpus of knowledge programmed into the deductive system. The first applications will probably be concerned with instructing a robot and with analysis of children's stories at the first-grade levels.

Loosely Stated Mathematical Problems

Several years ago, Daniel G. Bobrow completed a program that was able to solve some algebra problems stated in ordinary English. The mathematical material was rather sharply restricted to converting the sentences into simultaneous linear equations. Recently, Eugene Charniak has completed a new program that can solve some problems like this, stated in imprecise English:

> A train, starting at 11:00 a.m., travels east at 45 miles per hour while another, starting at noon from the same point, travels south at 60 miles per hour. How fast are they separating at 3:00 p.m.?

Solution of this problem requires a number of intellectual skills. Among these are, of course, the ability to manipulate the English text to derive a well-formulated symbolic problem, and knowledge of elementary calculus necessary to solve the symbolic problem. Perhaps less obvious, but equally important, is access to miscellaneous knowledge about the real world: for example, the knowledge that east and south are orthogonal directions, and enough knowledge about time to deduce that 3:00 p.m. is four hours later than 11:00 a.m.

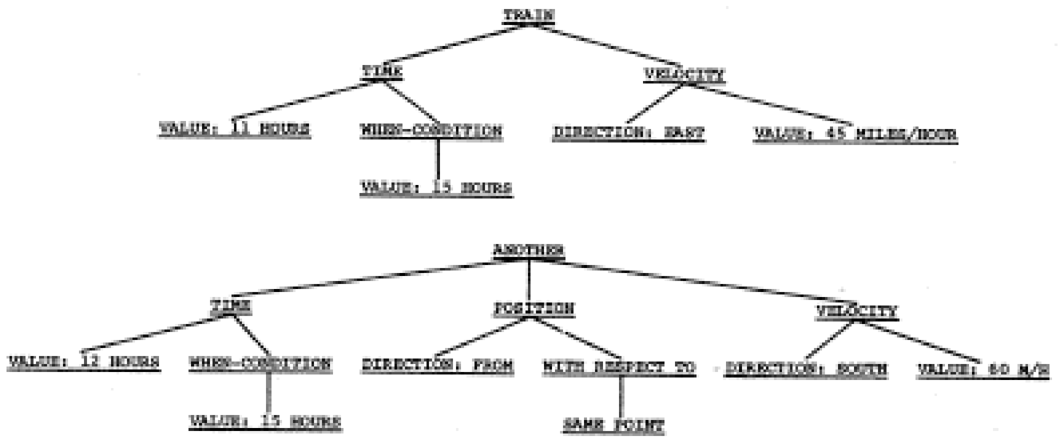


Fig. 29

Charniak's system, CARPS (CAlculus Rate Problem Solver) (MAC TR-51), translates this problem into an internal representation of which a general impression can be gleaned from Fig. 29. This structure indicates two objects, TRAIN and ANOTHER. Associated with each is a velocity given as direction and magnitude, a TIME that has a starting value for each train and a WHEN-CONDITION. The latter refers to the fact that most calculus "rate problems" ask questions roughly of the form, "What is A when B?"; in our case, we obtain, "What is the speed at which the trains are separating when the time is 3:00 p.m.?"

CARPS uses this information structure to formulate an algebraic manipulation problem. HOW FAST and SEPARATING indicate that what is desired is the rate of change of the distance between the two objects. The distance is the hypotenuse of a right triangle with legs toward EAST and SOUTH. The lengths of the legs are obtained by multiplying the rate of travel by the time. The expression for the derivative is therefore

$$\frac{d}{dt}\sqrt{(45[t-11])^2 + (60\ [t-12])^2}\,.$$

The desired value is at 3:00 p.m. (our WHEN-CONDITION). The derivative is found by algebraic manipulation, and the program then types

THE ANSWER IS 72.246 MILES/HOUR.

For its algebraic operations, CARPS uses parts of the MATHLAB programs described elsewhere in this report. CARPS has been used to solve other problems taken verbatim from calculus textbooks. These problems deal with cones, spheres and shadows as well as distances. In each case, the program was given sufficient knowledge to parse the sentences, set up an internal structure, and generate the equations.

We do not wish to imply that the program is very strong at solving calculus problems. Frequently, a problem cannot be solved because the program is unable to handle the syntax used, no method of solution is known to the program, or the problem requires facts about the real world which the program does not know or could not handle.

An example of the last difficulty arises in:

> A ladder 20 feet long leans against a house. Find the rate at which the top of the ladder is moving downward if its foot is 12 feet from the house and moving away at the rate of 2 feet per second.

Most adults have little difficulty in visualizing the situation described in this problem. CARPS is stuck because it does not know in which direction the foot of the ladder is moving. The phrase MOVING AWAY is interpreted by people to mean "moving away along the horizontal ground on which the foot of the ladder is presumed to rest". CARPS, however, does not realize that.

Clearly, CARPS's lack of such real-world knowledge cannot be circumvented or ignored. A crucial part of research towards problem-solving ability of this sort is concerned with the representation and use of such knowledge.

We have discussed (A.I. Memo 185) preliminary studies attempting to analyze the sorts of knowledge used by a first-grade child in understanding children's stories. Much of the calculus student's "general knowledge" is already structured at the elementary-school level, and we feel that progress in this area is essential to the future development of anything like "general intelligence". We have decided to make this area a very large part of our work in the next few years.

MATHLAB

MATHLAB is an interactive computer-program system that is being developed to facilitate creative work that involves extensive manipulation of symbolic algebraic expressions. MATHLAB is intended to be of help to research mathematicians and to appliers of mathematics engaged in "heavy manipulative work". In the MATHLAB project, therefore, serious attention has been paid to human engineering and to efficiency and speed of operation as well as to basic mathematical problems and to the problem of programming the computer to exercise initiative and some judgment in selecting and carrying out transformations of algebraic expressions.

MATHLAB is our best illustration of the idea of building "knowledge" into computer programs. The MATHLAB programs are actually quite capable in solving certain non-trivial mathematical problems. The essential idea, however, is to include provisions for interaction with that knowledge so that the user can build and administer complex but well-understood procedures. At best, the user and the programs supplement and reinforce one another and march rapidly through symbolic-manipulation problems that would take the unaided mathematician countless hours.

This last year, a new and much advanced MATHLAB was planned and partly implemented. Progress was made on faster parsing algorithms and on a representation of polynomials that speeds up addition and multiplication. At the same time, a significant advance in programmed symbolic integration was achieved by implementing a decision procedure, due to Risch, for expressions involving rational functions, logarithms and exponentials.

Also this last year, an extension of MATHLAB to handle special functions defined as integrals has enabled Moses to verify or correct some published tables of integrals. In 1958, W. D. Maurer of the Argonne National Laboratory compiled a table of 150 integrals involving the error function erf(s). Maurer was unable to verify, by hand, the following integral, which he included in the compilation with a note to that effect. In fact, it was slightly incorrect as printed:

$$\int x^2 \operatorname{erf}(ax + b) e^{p \cdot x} dx = \frac{1}{p^3} \operatorname{erf}(ax + b) e^{px} (2 - 2px + p^2x^2)$$

$$+ \frac{pax - pb + p^2/2a - 2a}{a^2p^2\sqrt{\pi}} e^{-(ax+b)^2+px}$$

$$- \frac{p^4/a - 4p^3b + 4ab^2p^2 - 2ap^2 + 8a^2bp + 8a^3}{4a^3p^3} e^{-p/a(\frac{p}{4a} - b)} \operatorname{erf}(ax + b - \frac{p}{2a}).$$

Moses's program was able to find the error.

Here is an example of factorization of a polynomial:

```
(1)  x**6 - 1 = 0                        (typed in)
```

(2) $x^6 - 1 = 0$ (MATHLAB's response).

In line 1, the user typed an equation. (The syntax is awkward because the typewriter is a one-dimensional device.) Line 2 shows the computer's response displayed in the conventional two-dimensional syntax of mathematics.

```
(3)  'PF('WS)                            (input)
```

(4) $(x-1)(x+1)(x^2+x+1)(x^2-x+1)$ (response).

In line 3, the user asked the system to factor the equation in line 2 by requesting the operation PF (Polynomial Factorization) to be applied to the WS (Work Space). The Work Space is always the last expression known to the system. In the response given in line 4, the quadratics are not fully factored because the program is restricted to finding the smallest factors that have integer coefficients.

Next we shall ask the system to integrate an expression, differentiate the result, and then check to see whether or not the result is identical with the original expression.

```
(5)  1/(x**3 + A*x**2 + x)               (input)
```

(6) $\dfrac{1}{x^3 + Ax^2 + x}$ (response)

```
(7)  'integrate('WS, x)                  (input)

(8)  IS THE EXPRESSION
```

$A^2 - 4$

to be considered positive, negative or zero (response)

```
(9)  NEGATIVE                            (input).
```

The computer wants to avoid terms in the integral which have complex values, if possible. This is the reason for the question posed in line 8. Line 9 is the user's response, and line 10 is the integral. Obtaining the result involves use of a subsystem for handling partial fraction decompositions and one for integration of rational functions:

(10) $- 1/2 \log(x^2 + Ax + 1)$

$$+ \frac{-A}{\mathrm{sqrt}(-A^2+4)} \arctan\frac{2x + A}{\mathrm{sqrt}(-A^2+4)} + \log(x).$$

We shall now differentiate the integral.

(11) 'DERIV('WS, x) (input)

(12) $$\frac{-\ 2A}{\left(\frac{(2x+A)^2}{-A^2+4} + 1\right)(-A^2+4)} + \frac{-1/2(2x+A)}{x^2+xA+1} + \frac{1}{x}.$$

Well, the result is certainly not identical with our original expression (line 6). This is because the differentiation program differentiates a sum term-by-term without combining the results. (Note that the log x term in line 10 gave rise to the term in line 12.) To our rescue comes a simplification program RATSIMP (RATional SIMplification) which will expand denominators and combine the entire result into a single fraction. This fraction is simplified by removing the greatest common divisor of the numerator and denominator.

(13) 'RATSIMP('WS,x) (input)

(14) $\frac{1}{x^3 + Ax^2 + x}$ (response).

Here is another integration problem:

(15) x**3/(1 - x**2)**(3/2) (input)

(16) $\frac{x^3}{(1-x^2)^{3/2}}$ (response)

(17) 'integrate('WS, x) (input)

(18) $\mathrm{sqrt}(-x^2 + 1) + \frac{1}{\mathrm{sqrt}(-x^2 + 1)}$ (response).

Note the system's preference for $-x^2 + 1$ over the more conventional $1 - x^2$. Mathematicians today tend to obey the rule: If a sum of two terms contains a positive term and a negative term, write the positive term first. Graphic heuristics of this sort are now included in Martin's more sophisticated display routines.

(19) (2*x**6+5*x**4+x**3+4*x**2+1)/(x**2+1)**2*e**x**2

(input)

(20) $\frac{2x^6 + 5x^4 + x^3 + 4x^2 + 1}{(x^2 + 1)^2}e^{x^2}$ (response)

(21) 'integrate('WS, x) (input)

(22) $\frac{2x + 2x^3 + 1}{2x^2 + 2}e^{x^2}$ (response).

These results depend on a powerful pattern-matching program for algebraic expressions.

This example points clearly the need for a better way to enter mathematical expressions into the system. Methods that allow users to hand-write algebraic expressions, using a pen-like device, are now close to the step of useful application (Ref. 7).

The following example shows the system solving a linear differential equation with constant coefficients, a problem-type of great interest to electrical engineers.

```
(23)  DERIV(y,x,3) + A*DERIV(y,x) = sin(2*x)
                                              (input)
```

(24) $\frac{D^3y}{Dx^3} + A\frac{Dy}{Dx} = \sin(2x)$ (response)

```
(25)  'LDESOLVE('WS, y, x)                    (input)

(26)  NEED INITIAL CONDITIONS                 (response).
```

The program allows one either to enter specific initial values for Y(0), Y'(0), and Y"(0) or to leave them as indeterminates (as is done below).

```
(27)  ALLFORMAL                               (input)

(28)  IS THE EXPRESSION

      A

      TO BE CONSIDERED POSITIVE, NEGATIVE OR ZERO

                                              (response).
```

As before, the answer to this question will be used to generate an answer which contains no complex terms.

```
(29)  POSITIVE                                (input)
```

(30) $$\frac{2Y(0) + 2Y''(0) + 1}{2A} + \frac{-Y''(0)A + 4Y''(0) + 2}{A^2 - 4A}\cos(\mathrm{sqrt}(A)x) + Y'(0)\,\frac{\sin(\mathrm{sqrt}(A)x)}{\mathrm{sqrt}(A)} + \frac{-1}{2A - 8}\cos(2x)$$
(response).

The solution is obtained with the help of a subsystem that takes Laplace transforms of both sides and obtains the inverse Laplace transform of their ratio, using the package for integrating rational functions.

Hierarchical organizations such as MATHLAB's will become increasingly popular as programs and systems get more complex and sophisticated. While this system is ostensibly working at the higher levels of the hierarchy it is actually spending most of its time at the lower supporting levels. Probably, over half the routines of its more than 60,000 words of memory are being utilized in solving a differential equation. This should serve as a warning to designers of time-sharing systems who would prefer that programs limit their use of memory.

The field of computer-aided instruction (CAI) has not progressed to the extent that many have wished. Teaching programs are unable to answer questions other than those that the designer had foreseen -- not only because these programs do not know enough English to understand the question, but basically because they do not at all understand the field about which they are supposed to teach. The designer of the usual kind of CAI course writes a script which is controlled by a context-independent interpreter. The script designer must provide for many possibilities at each step because he cannot rely on the interpreter of the script to know anything about the field with which the script deals. If a student is allowed, as he rarely is, a reasonable flexibility in replying to the program, then either the script designer is physically exhausted from considering all the plausible replies, or he misses considering many such replies. We contend that a primary step in writing a sophisticated teaching program should be the education of the program in the area about which it must teach.

Consider, for example, the problem of writing a program for the teaching of freshman calculus techniques for differentiation and integration. Using the knowledge of these techniques that is embedded in the MATHLAB system, it would be possible to write teaching programs that check the steps of a student's calculation, advising him of errors as he progressed. A preliminary program with these facilities has been prepared. Such a program could also be capable of performing differentiation or integration problems suggested by the students, explaining its steps as it proceeded. To achieve such capabilities through the writing of mindless scripts is unthinkable. To be sure, the experiments with the Calculus Rate Problem Solver (CARPS) have shown that a great deal more must be known before such programs can deal effectively with human beings in a wide area of knowledge.

The task of finding out what human users know is clearly related to the tasks of workers in the field of linguistics, psychology, and philosophy. A major failing, however, in the education given in these traditional disciplines is the lack of understanding of the concept of an algorithm or process. We believe that advances in our understanding of human knowledge as algorithmic processes are likely to be of supreme importance.

Although the most essential part of MATHLAB is conceptual and the next most essential is the programming of symbolic transformations, the interface with the user is very important, and we look forward to giving MATHLAB every advantage of effective man-computer-interaction techniques.

Chess

MACHAC-VI, a chess program developed by Richard D. Greenblatt for the PDP-6 computer, was begun in November 1966. It entered its first tournament the following January. Since then, it has participated in four more official U.S. Chess Federation tournaments and has achieved an official rating of 1528. In its last tournament, it had a performance rating of 1720 and drew an 1880 player. These statistics put it a little more than one standard deviation below the mean strength of all U.S. tournament players (which is about 1880). The machine has played perhaps 2000 games against chess players of every strength. It wins about 86 per cent of its games against tournament players. Greenblatt estimates that it represents a programming effort of about six man-months.

Once the basic program was completed, the first step in refining its play was to give it a model of the flow of power on the chess board. Next, it was given explicit techniques for pins and discoveries; then it was given positional and pawn structure considerations. These and other features are largely completed now.

The next major step, only partially implemented now, will be to incorporate symbolic reasoning ability into the program. The opponent's threats are specifically identified, and each reply is evaluated for its ability to answer some or all of the threats. This does not mean that the opponent's threats were ignored before; but now the program proceeds with a more explicit sense of what it is doing, whereas in the classical minimax strategy the threats are implicit and therefore cannot be subject to symbolic statements of what to do. The explicit system should be much more efficient.

Another new feature, already implemented, adapts the plausible-move system to recognized states of the game: if one side appears to be ahead in the look-ahead analysis, it will prefer holding, trading and simplifying moves; the other side will require positive, attacking moves that might yield tangible gain. These and other features are operational, but the other programs have not been modified to take as much advantage of them as they probably could.

Another class of improvements is planned which will include real symbolic learning rather than mere tuning-up. We expect that these, with improvements in end-game strategy, will result in a substantial increase in strength of play.

Eye-Tracking

The much-used term "man-machine interaction" usually refers to situations in which the communication relationship is exceedingly lopsided. Man can see, hear and touch the computer in many ways and places, but the machine is restricted to receiving information through a narrow bottleneck -- usually a Teletype. We have been interested for a long time in redressing this imbalance. This year we were able to achieve an old goal of enabling the machine to look at a person. More precisely: the machine looks at the man's eyes to determine his point of fixation.

Recording eye movements is a well-established technique in the study of perception, tracking skills, and even problem-solving behavior. But traditional methods give the pattern of eye movement only after analysis. We believe our system, which incorporates an eye-tracking device developed by J. Merchant of Honeywell under NASA contract, is the first that enables a computer to use the information in real time. As an example of its uses, Samuel L. Geffner has two programs that display text for a subject (such as a small child) to read and take action related to the word currently fixated. One of the programs causes the word to be pronounced by the computer; the other causes the word to be replaced, in the display, by its translation into another language. These programs are mere demonstrations -- but they illustrate rich applications to teaching and other interactive situations. We are pursuing such applications, partly with support from NASA.

The A.I. Time-Sharing System

The experimental work of the Artificial Intelligence Group requires a high level of computer service for a limited number of users. Our system is based on time-sharing a two-processor machine (PDP-6 and PDP-10) with a core memory of 2^{18} words of 36 bits. Normally, the programs of the active users remain in fast memory; this limits the number of users now, but a paging device to be completed early in 1970 should allow some expansion required by the maturation of certain projects, notably MATHLAB.

The special requirements of the project led to a time-sharing system with enough novel features to merit discussion. Donald E. Eastlake describes the system in detail ("ITS 1.5 Reference Manual", A.I. Memo 161a). Besides the usual kinds of input-output and system calls, there are special calls to reduce overhead and to facilitate real-time control. These include programs for operating the mechanical hands and computer eyes and other special remote-control devices. All ordinary time-shared programs run in one of the processors, and critical real-time processes are assigned, by user calls, to the other. The real-time processor is normally assigned to a single user at a time.

Because all user programs ordinarily reside in core, it is possible to switch between programs with great rapidity. Quanta of user time are short enough to allow program response to single typed characters without noticeable delays. When swapping is introduced, we expect it to affect only semi-dormant users.

A user may have many jobs running "simultaneously". Each user commands a tree of procedures; each can create and control subordinate procedures; all have equal access to external devices and files. The top procedure, loaded automatically when the user declares his existence, contains a version of the well-known DDT debugging system. A special character allows transfer to the top procedure from any level, so that the user is automatically in position to use DDT's interrogation, breakpoint, and other debugging features.

Input-output devices are referenced symbolically and data can be transferred on character, word or block bases; an entire video image can be acquired by a system call while the calling procedure continues without waiting. User programs need no buffers in their own core images. Line-printer requests, for example, are buffered until the device is free. User programs communicate by "software interrupts" that are treated the same as hardware interrupts; superior procedures can store and retrieve words in inferior procedures as though they were I/O devices; buffered communication is provided so that pairs of procedures can treat each other as input devices. The file system resembles the CTSS file system.

The schedule algorithm tries first to equalize time between users, and second to equalize times between the procedures of a single user tree. A special procedure always runs which performs various jobs and can check constant portions of the system against a copy in an attempt to detect some forms of hardware or system failure.

The secondary storage uses IBM 2311 discs and DEC microtape drives which are file-structured in the same way. Users can establish symbolic links between file directories so that files can be shared by many programs and many users; these links can be chained.

The system has several operating stations. These include four text-display devices, several Teletypes and external telephone lines, a main console with DEC 340 display and several slave monitors around the laboratory, and a special console with controls for operating the eye-hand system. The system interfaces with a radio transmitter for inexpensive long-distance remote experiments.

A multiplexed digital-analog system operates either in word or block mode, on call or automatically, as requested. Any number of users may simultaneously have analog access on different channels in different modes. These are some of the real-time facilities:

> 1) Iris, focus, stereo mirror, for high-resolution image dissector,
>
> 2) Pan, tilt, zoom, focus, iris, for small, low-resolution image dissector,
>
> 3) Extend, tilt, rotate, grasp, finger curl, for hand MA-3,
>
> 4) Extend, tilt, rotate, grasp, for hand MA-2,
>
> 5) Roll, yaw, horizontal, vertical, swing, for arm MA-2,
>
> 6) Position for tactile-sensor device,
>
> 7) X, Y, Z, rotate for arm MA-3.

For moving the arms, special system calls provide acceleration- and velocity-limiting, software limit stops and other performance limits. To prevent disaster, certain motion commands are uninterruptable for limited times, and an arm-moving call is illegal if the device is under control of another user. Motion calls are not automatically buffered like those to the line printer! Although one can read any input channel at any time, most mechanical devices can be operated at leisurely speeds, with the input multiplexer channels read automatically. The system normally does this at a minimum of 5 times per second.

On the input side are sensors for all mechanical degrees of freedom, including hands and arms and optical parameters. There is a test stand for calibrating the positional control of the arms, and there is a variety of portable remote-control boxes with switches and continuous-adjustment knobs. A special system call gives the user great flexibility in real-time control of program parameters. It connects potentiometers (through the input multiplexer) to arbitrary program variables -- i.e., memory registers -- and these can be specified to be fixed or floating, or arbitrary bytes. There are options for assigning absolute or incremental meanings to knob positions; in the incremental mode, there is a programmed velocity-dependent gain and a side-to-side hysteresis so that it is easy to make fine and coarse adjustments with the same control, and it tends to remain centered in its motion range.

The system contains good facilities for graphic display, which are presently limited to one user. Growing needs dictate acquisition or multiple-display hardware.

LISP (MACLISP)

LISP is the high-level language the Artificial Intelligence Group uses. Our LISP version is different in many ways from other LISPs, and some of these differences represent progress. The details are of interest only to specialists, who may consult the memo by Jon L. White ("Time-Sharing LISP for the PDP-6", A.I. Memo 190). The system does not use an A-list for ordinary variable bindings; it uses a value property; a special stack is used to unbind after lambda-conversions. There are many small improvements in human engineering such as default values of non-specified function arguments. The garbage collector manages space for arrays. The READ program allows new character-handling facilities and macro definitions of certain kinds. There is a variety of powerful editing facilities and display packages, and some strong debugging systems. The machine-language feature, LAP, has been strengthened. Whitfield Duffie has restructured the compiler to make the most of its decisions by dispatching on tables so the users can introduce special forms by giving the compiler instructions through the table entries. It would be more efficient if the compiler were able to make a deeper and more systematic analysis of the uses of expressions being compiled, separating executions for value from those for effect on flow of control, for side effects, etc., and for recognition of unnecessary recursion and similar simplifications. This itself is a problem in artificial intelligence.

Motivated by MATHLAB and other projects, we are now making an effort to introduce facilities for very fast arithmetic into LISP. We expect that the resulting system will be almost as efficient at number-crunching computation as is any conventional algorithmic language.

Mechanical Hands and Arms

The project has developed several mechanical effector devices for computer-controlled manipulation. This field is still substantially unexplored, and our work can be considered only to clarify some of the problems, not to solve many. There are problems in several different areas.

There is a need for much deeper analysis of what is needed in designing a hand-arm system for various kinds of jobs. In his thesis, David L. Waltz studies motions the human hand uses to perform several basic tool-handling and similar tasks, to see which of the degrees of freedom of the hand were most essential, and how they were sequenced. There is some information about this in the orthopedic literature, but it is not sufficiently structured to serve as a base for programs. One needs to describe actions in terms of interactions of position, force, velocity and sensory responses -- in short, one needs something like a programming language for it. We also examined some choreographic languages but, although they contain some good ideas about path-of-motion description, they do not have sensory-interaction predicates. Waltz developed some notation that seems helpful and Ernst's (Ref. 8) discussion is still relevant. Waltz's (earlier) thesis shows how a few degrees of freedom can accomplish much, but one wishes there were much more known about this subject.

In most of our experimental work, we have employed a modified AMF Versatran industrial manipulator arm (which operates in cylindrical-polar coordinates). Although the hands we attached to this device have some extra motions, the large-scale motion of that type of arm has no such redundancy; thus there is basically only one way to reach each point in space. Calculating this is straightforward, solving a not-too-complicated equation that has only one solution.

Because this is a very clumsy device -- such an arm is unable to reach around obstacles or even to support an unobstructed object from an arbitrary direction -- we developed a much more mobile arm with many extra degrees of freedom. The new arm behaves more like a tentacle than like a coordinate system. Hydraulically operated, it is relatively slim and has five articulations, each with two degrees of freedom, a shoulder, three elbows and a wrist. An articulated prototype shows clearly that this geometry is adequate for everything a human arm can do and more. In the actual hardware mechanism, each of the eight interior hinge joints has approximately 110 degrees of flexion. We conclude that this is inadequate; it must be more nearly 180. The lesson we learned is that in a minimal system a small angular restriction means only that the work space is somewhat reduced. But, in a linkage whose purpose is a great variety of ways to reach points in the work space, all restrictions interact in messy ways to break up the higher-dimensional mobility space into hard-to-understand fragments, so that two slightly different three-dimensional positions may have to be reached (if at all) by grossly different global arm configurations. For such a mechanism with 10 degrees of freedom for reaching points in only three dimensions, there is a mathematical problem of inverting the position equations. One can sometimes get by with simple hill-climbing by successive approximations, but this is really unsatisfactory because it is hard to adjoin global information, and it does not give an analytic picture of the alternative solutions to the problem. Jean-Yves Gresser, in his (earlier) thesis discusses a method that gives a solution specific to our arm-mechanism, using a rather general method of dividing the position problem into a series of almost-independent factor sub-problems. What Gresser does is to divide the arm in half, and describe the mobility space of each half, using simplifying approximations for each side. When the half-solutions are put together, there needs to be an accuracy-increasing iteration anyway, so there is hardly any real loss through approximating.

Now, in our arm-mechanism, it happens that the mobility zones of the half-arms can be described in terms understandable to humans -- for this particular arm, each zone resembles a portion of a torus. To find the ways to get the whole arm to a certain space position, one describes the intersection of the two tori, one centered at the shoulder position and the other the goal position. Each intersection point yields an approximate solution to the problem (that is, a possible location of the center of the arm), and a higher-level program could be asked which solution is most desirable.

There is a larger problem here: how should one represent a machine's body image? For the problem of a single, not-too-complicated arm, one can doubtless get by with cleverly coded, sparse, three-dimensional arrays, but one would like something more symbolic. And one wonders what happens in the nervous system; we have not seen anything that might be considered to be a serious theory. Consider that a normal human can place an object on a table, turn about and make a gross change in his position and posture, and then reach out and grasp within one or two inches of the object, all with his eyes closed! It seems unlikely that his cerebellum could perform the appropriate vector calculations to do this; it is not a mere matrix inversion (and, even if it were, one would still need a theory). We would presume that this complex motor activity is made up, somehow, of a large library of stereotypical programs, with some heuristic interpolation scheme that fits the required action to some collection of reasonably similar stored actions. But we have found nowhere any serious proposal about neurological mechanisms for this, and one can only hope that some plausible ideas will come out of robotics research itself. Unfortunately, at present this area is somewhat dormant.

Another conclusion from our experiments is that delicate manipulations must be controlled by application of controlled forces (rather than by direct position control), with attention to matching velocities with inertias. For measuring the forces on a mechanical hand, we have taken two approaches. First, it is quite feasible to engineer a grasping surface with good pressure sensitivity at a great many points by wrapping a coaxial cable connected to a Time-Domain Reflectometer. (A TDR is a sort of radar system designed to measure reflected radio waves that are produced in a soft-sheathed tube, by any deformations of the wall. The instrument permits hundreds of thousands of points, using a single electrical connection to the hand. (Waltz describes this system in his thesis.) Second, we have developed a strain-gauge telemetered wrist that provides force information in the necessary six degrees of freedom. This little instrument is able to tell where and in what direction the hand is being pushed, assuming that the force is being applied at a single point -- a condition that usually holds in a first contact with an object.

Solution of the appropriate moment equations for the set of six forces at the wrist (Fig. 30) yields a certain line in three-dimensional space, along which a force of a certain magnitude is applied. If the machine knows the geometry of its hand, it can presume that the force is applied where this line intersects the hand.

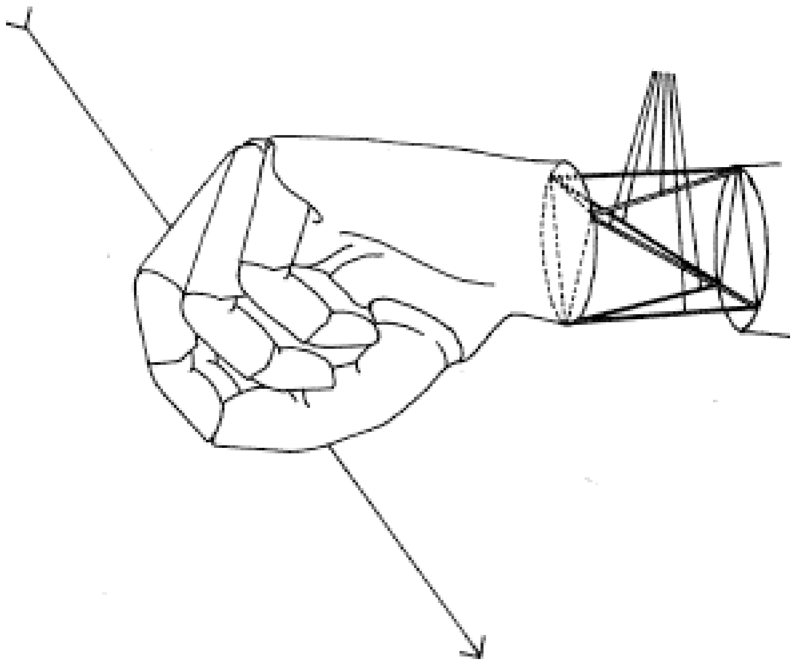


Fig. 30

We have built this hand and are testing it. We shall describe it in a memo by Callahan, Shah and Minsky.

Computer Eyes

A computer must have eyes if it is to see. When the project started, no device was available for interfacing a computer with a real-time camera, although there were film-reading devices on the market. We first used a Vidicon television camera, but we had difficulties with the limited signal-to-noise quality of the Vidicon. Besides, Vidicons are not suitably adaptable to random-access scanning, they have motion persistence and other problems. We decided that this added up to too many obstacles. The image-dissector camera offered much cleaner images (at the expense of sensitivity, which, for laboratory purposes, was unimportant) so, in collaboration with Information International, Inc., we designed and constructed our present camera system. An image dissector is essentially an inverted cathode-ray tube: an image projected on a photo-cathode produces electrons which are focused through a deflection system so that the current from a selected image point falls through a small hole and is then measured.

In such a system, the signal-to-noise ratio depends essentially on the number of electrons, and one can obtain more precision at the price of longer time-exposures. We decided to put this signal-to-noise ratio under direct program control and to make it independent of the brightness of the point being measured. This is built into the camera's video processor. Horn gives the details ("The Image Dissector 'Eyes'", A.I. Memo 178). If the constant signal-to-noise constraint were enforced even on very dim points, the exposure times would become excessive, so the system has also a dark cut-off parameter that terminates the measurement very early if an initial estimate of the brightness falls below a specified value. This is also built in the video hardware. The resulting system can measure intensity over a 4096:1 dynamic range, with a brightness-discrimination sensitivity of one part in 64 throughout this range. The output appears as a floating point number or as a logarithm of intensity. The eye has also a reference-brightness input that can be used to make the measurements almost independent of overall illumination fluctuations, and there are various protective devices to prevent damage to the image tube by excessively bright light sources. Horn's report discusses many advantages and pitfalls in using this system; anyone contemplating using image dissectors for computer vision should correspond with us about the results of some modifications now under construction.

It is frequently suggested that one ought to build various forms of parallel-processing artificial retinas, perhaps along the lines of biological systems. But the operations of the vertebrate retina and the subsequent image processing is not nearly so well understood as is generally believed, and we do not think the time is quite ripe for taking such a step in hardware. In our present operations, the computation time consumed in the higher levels of scene-analysis is comparable to that spent in the lower-level pre-processing, so at present there would be no large time factor to be gained from fast parallel hardware -- even if we were able to decide how it should work.

References

1. Lawrence G. Roberts, Machine Perception of Three-Dimensional Solids, Department of Electrical Engineering, M.I.T., Ph.D. Thesis, May 1963.

2. Thomas G. Evans, "A Program for the Solution of Geometric-Analogy Intelligence Test Questions", Semantic Information Processing, M.I.T. Press (Cambridge) 1968, pp. 271-353.

3. A. Newell, "Learning, Generality and Problem Solving", Information Processing 1962, IFIP Congress 1962, North Holland Publishing Co. (Amsterdam) 1963, pp. 407-412.

4. Edward A. Feigenbaum, "The Simulation of Verbal Learning Behavior", Computers and Thought, McGraw-Hill Book Co. (New York), 1963, pp. 297-309.

5. M. A. K. Halliday, "Some Notes on 'Deep' Grammar", J. Linguistics, 3, 1967.

6. Terry A. Winograd, "Linguistics and the Computer Analysis of Tonal Harmony", J. Music Theory, 12, 1968.

7. R. Anderson, "Syntax-Directed Recognition of Hand-Printed Two-Dimensional Mathematics", Ph.D. Thesis, Division of Engineering and Applied Physics, Applied Math, Harvard University, Cambridge, Mass., Jan. 1968.

8. H. A. Ernst, MH-1, A Computer-Operated Mechanical Hand, Department of Electrical Engineering, M.I.T., Ph.D. Thesis, December 1961.

Some Relevant Memos of the A.I. Group

101 SIDES 21, R. D. Greenblatt, J. T. Holloway; described by Donald Sordillo, August 1966, MAC-M-320.

123 Computer Tracking of Eye Motions, M. L. Minsky, S. A. Papert, March 1967 (Vision).

131 POLYSEG, A. K. Griffith, April 1967 (Vision).

145 A Fast-Parsing Scheme for Hand-Printed Mathematical Expressions, W. A. Martin, Oct. 1967, MAC-M-360.

157 Time-Sharing LISP for the PDP-6, John White, March 1968. Replaced by Memo 190.

160 Focusing, B. K. P. Horn, May 1968.

161A ITS 1.5 Reference Manual, D. E. Eastlake III, MAC-M-377. Revised July 1969 (ITS 1.4 Ref. Manual June 1968).

163 Holes, P. H. Winston, Aug. 1968.

164 Producing Memos using TJ6, TECO and the Type 37 Teletype, L. J. Krakauer, Sept. 1968. Replaced by Memo 164A.

165 Description and Control of Manipulation by Computer-Controlled Arm, J.-Y. Gresser, Sept. 1968.

166 Recognition of Topological Invariants by Modular Arrays, W. T. Beyer, Sept. 1968.

168 PLANNER, C. E. Hewitt, MAC-M-386. Oct. 1968; revised June 1969.

169 PEEK and LOCK, D. E. Eastlake III, MAC-M-387, Nov. 1968. Replaced by revised Memo 161A, July 1969.

171 Decomposition of a Visual Scene into Three-Dimensional Bodies, Adolfo Guzman, Jan. 1961, MAC-M-391.

172 Robot Utility Functions, Stewart Nelson, Michael Levitt, Feb. 1969 replaced by Revised Memo 161A, July 1969.

173 A Heuristic Program that Constructs Decision Trees, March 1969, P. H. Winston.

174 The Greenblatt Chess Program, R. D. Greenblatt, D. E. Eastlake III, Stephen Crocker, April 1969.

175 On Optimum Recognition Error and Reject Tradeoff, C. K. Chow, April 1969.

176 Discovering Good Regions for Teitelman's Character Recognition Scheme, P. H. Winston, May 1969.

177 Preprocessor for Programs Which Recognize Scenes, H. N. Mahabala, August 1969.

178 The Image Dissector "Eyes", B. K. P. Horn, August 1969.

180 The Integration of a Class of Special Functions with the Risch Algorithm, Joel Moses, MAC-M-421.

181 PROGRAMMAR: A Language for Writing Grammars, Terry Winograd, November 1969.

182 Display Functions in LISP, Thomas Binford, 1970.

183 On Boundary Detection, Annette Herskovits and Thomas Binford, July 1970.

185 ARPA/ONR Proposal VI December 1970 through November 1971, M. Minsky and S. Papert.

ARTIFICIAL INTELLIGENCE

July 1967 – June 1968

Academic Staff

M. Blum
M. L. Minsky
W. A. Martin
J. Moses
S. A. Papert

Non-Academic Research Staff

R. J. Abbott
G. W. Baylor
M. D. Beeler
W. Bennett
T. O. Binford
R. W. Gosper
R. D. Greenblatt
R. W. Henneman
P. Himot
J. T. Holloway
R. Noftsker
J. S. Roe
P. R. Samson
C. Spielman
G. Voyat
J. L. White

Instructors, Research Associates, Research Assistants and Others

R. M. Baecker
W. T. Beyer
E. Charniak
S. D. Crocker
S. L. Geffner
A. K. Griffith
A. Guzmán
C. E. Hewitt
T. L. Jones
L. J. Krakauer
M. J. Lennon
M. E. Manove
G. H. Mitchell
D. N. Perkins, Jr.
C. R. Smith
S. W. Smoliar
M. Speciner
G. J. Sussman
D. L. Waltz

Guests

A. Forte - Yale University
C. Engelman - MITRE Corporation
R. Silver - MITRE Corporation

Research on Intelligent Automata – Marvin Minsky and Seymour Papert

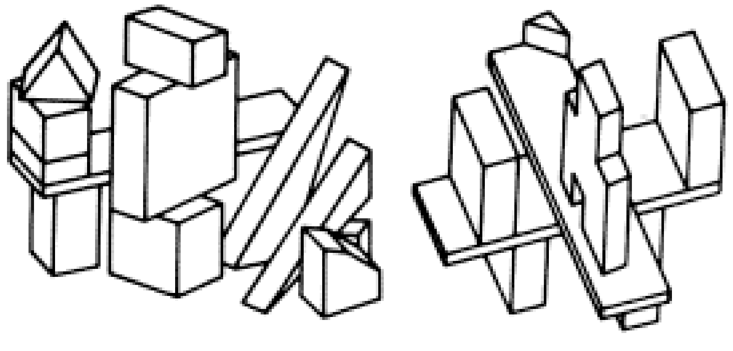
The largest sector of research in our group is still the study of methods for providing machines with greater visual and manipulative capabilities. Our general approach to this goal was described in last year's MAC Progress Report. We expect, some time in 1969, to demonstrate some practical capabilities of automatic, visually guided manipulation, by showing the computer a structure made of children's blocks, and having it build a functional duplicate. Details of this work are described separately in the Status Reports of the Intelligent Automata Project, and in many of the Artificial Intelligence group memoranda available through our office.

Computational Geometry – Marvin Minsky and Seymour Papert

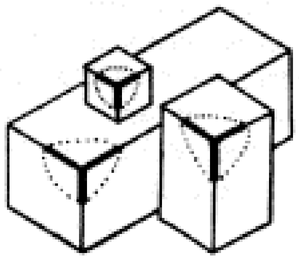
Preoccupation with theoretical aspects of machine vision has led us to crystallize a general concept we call "computational geometry". This is a new mathematical specialty concerned with the complexity of computations necessary to recognize various properties of geometric objects. The rapid evolution of discoveries in this area has given us hope that it will lead to new directions for "computer science" in general, by providing subject matter that combines intuitive clarity and practical importance, with close connections between classical parts of mathematics – geometry and algebra. We believe the widely recognized conceptual fragility of current "theories" about computation is due in large part to their attempt to build upon excessively abstract principles of automata theory and linguistic structure, without enough concern for thorough understanding of particular problem areas in relation to particular machine structures. The following sections show some examples.

A. The SEE Program

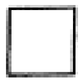
Computational requirements lead to geometric questions of an entirely new sort, such as surely never occurred to Euclid. Consider the need to analyze a scene in which some of the objects partially obscure others.



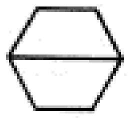
The methods described in last year's report presupposed a computer model or description of each kind of object – cube, pyramid, etc. Since then it has become clear, in further work of Adolfo Guzmán, that this information is not wholly necessary. Indeed, computations are usually much simpler if based on a more abstract and general theory of the appearances of objects. Earlier methods were based on partial recognition of the bodies such as the scene below:



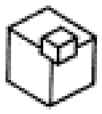
Here, where all objects are rectangular solids, and do not occlude one another badly, we can discover the objects by the extremely local process of locating all the "Y-joints." Each object contains at most one such distinctive feature. This could, of course, fail because of perspective, as in



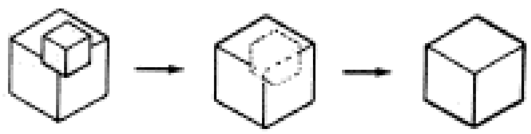
which could be a cube, or in



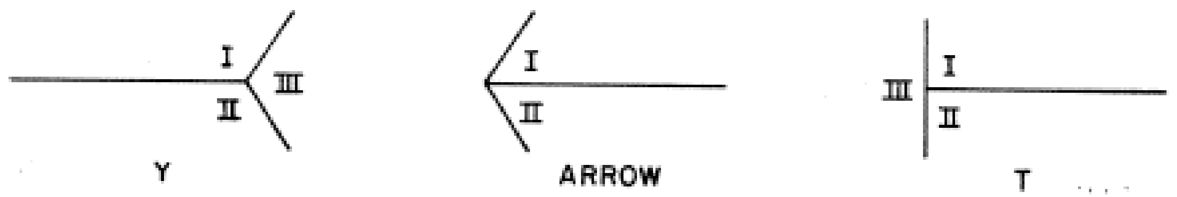
(for we require each of the three angles of a Y-joint to span less than 180 degrees). A more serious failure is in the case of occlusion, as in



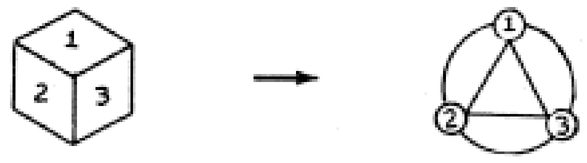
where one of the Y-joints is completely hidden from view. But the great power of programs capable of hierarchical decisions is illustrated by the possibility of first <u>recognizing the small cube</u>, then <u>removing it</u>, then <u>extending the hidden lines</u>, and so discovering the large cube!



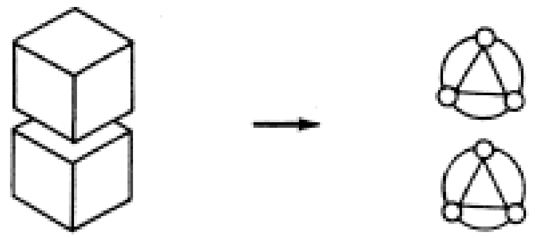
The program developed by Adolfo Guzmán proceeds in a rather different way; his idea is to treat different kinds of local configurations as providing different degrees of evidence for "linking" the faces that meet there. For example, in the following three types of vertex configurations



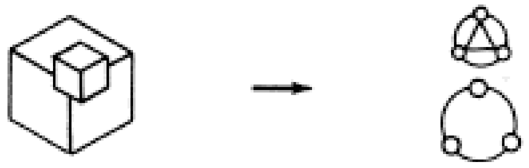
the "Y" provides evidence for linking I to II, II to III, and I to III. The "Arrow" just links I to II. Because a "T" is ordinarily the result of one object occluding part of another, it is not regarded as evidence linking I to III, or II to III (and it is also neutral about I and II). Using just these rules, we can convert pictures into associated groups of faces as follows: we represent Y-links by straight lines and arrow links by curves.



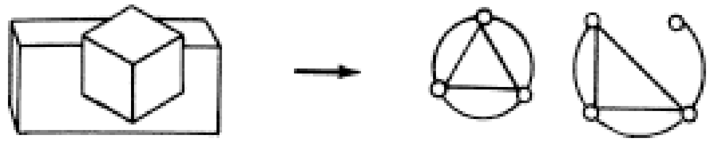
(1)



(2)

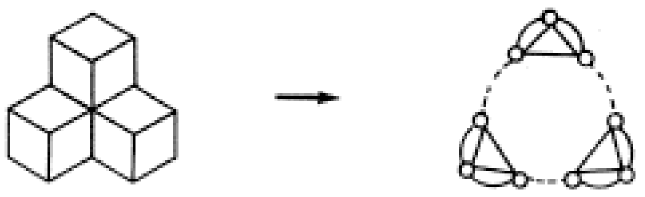


(3)



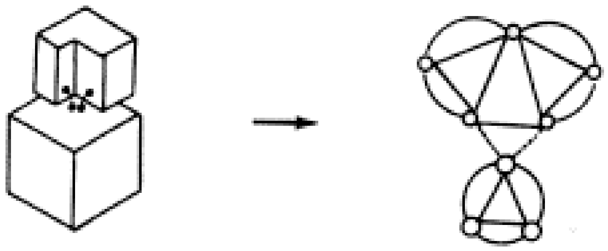
(4)

So far, there has been no difficulty in associating sets of linked faces with objects. The usefulness of the variety of kinds of evidence shows up only in more complicated cases. In the Example (5)



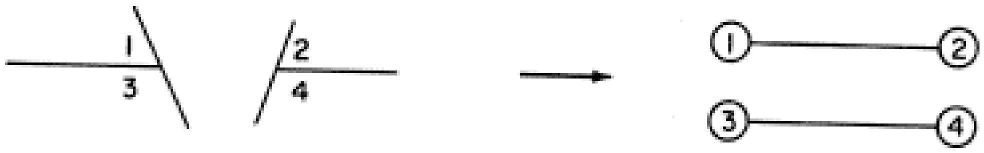
(5)

we find some "false" links due to the merging of vertices on different objects. To break such false connections, the program uses a hierarchical scheme that first finds subsets of faces that are very tightly linked (e.g., by two or more links). These "nuclei" then compete for more weakly linked faces. There is no competition in Examples (1)-(4), but in (5) the single false links between the cubes are broken by his procedure. In Example (6), if a very simple "competition" algorithm were not adequate here, one could also take into account the negative evidence the two T-joints provide against linking I-III and II-III.



(6)

We have described only the skeleton of his scheme; Guzmán uses several other kinds of links, including evidence from collinear T-joint lines of the form



and the effects of some vertices are modified by their associations with others. This variety of resources enables the program to solve complex scenes like that illustrated at the beginning of this section.

Full details will be in Guzmán's doctoral thesis to appear in the Spring of 1969. The surprising power and elegance of this algorithm suggests that there will evolve a rich theory of the geometry of object clusters. Further evidence for this comes from the discovery of simple heuristics that seem to generate plausible hypotheses about missing lines in pictures of complex scenes.

B. The Theory of Perceptrons

For several years, we have been interested in finding a theoretical basis for assessing abilities and limitations of the perceptron and similar machines. These are highly parallel computation schemes that attempt to recognize complicated inputs by 1) computing many properties of the input that are relatively easy to recognize, and then 2) basing a decision on some relatively simple combination of the results of the first stage, such as a comparison of weighted sums of evidence for each competing alternative possibility.

Our interest in such machines is not based on very much concern for their practical possibilities, for these are very limited. However, it is our conviction that these machines are nonetheless of critical importance as theoretical models, because if we cannot thoroughly understand such simple computers, we can have little hope of obtaining good theories of more powerful and general computers. (The "theory of computability" for "universal" machines is totally unsatisfactory in the context of any real practical problems.) Fortunately, we have obtained a wide variety of theoretical conclusions about perceptrons, and these are given in detail in a new book (Perceptrons: An Introduction to Computational Geometry, Marvin Minsky and Seymour Papert, M.I.T. Press, 1969).

We shall summarize some of the results here. First, let us define a perceptron of order K:

Let R be a part of a two-dimensional plane. Let X be an arbitrary black-and-white "picture", i.e., a subset of R. Any point in X is considered to be black, and the rest of R white. Let Φ be a set $\{\phi, \phi', \phi'', \ldots\}$ of predicates – functions whose values are 0 or 1 – each of which depends on no more than K points. Finally, choose for each ϕ a number α_ϕ and define

$$\begin{aligned}\psi(X) &= 1 \text{ if } \sum \alpha_\phi \phi(X) \leq 0 \\ &= 0 \text{ if } \sum \alpha_\phi \phi(X) < 0\end{aligned}$$

This definition includes the concept of "threshold", as in

$$\sum \alpha_\phi \phi(X) \leq \theta \qquad (\theta \text{ any real number})$$

if we allow one of the ϕ functions to be a constant.

Now we ask whether a perceptron can recognize a "pattern". For example is there a perceptron such that $\psi(X) = 1$ when X is a square (or convex, or connected) and $\psi(X) = 0$ when X is not a square (or not convex, or disconnected)? Our analytic methods depend mainly on replacing the geometric concept of a pattern by the algebraic properties of the transformations that preserve the features that concern us. We cannot give a full example of how this is done for geometric concepts, but the following sketch shows how the algebraic theory works in a fairly simple case. Chapter references point to corresponding sections in the book Perceptrons.

Theorem 3.1 (Chapter 3) Informal Version

Suppose the retina R has a finite number of points. Then there is no perceptron $\sum \alpha_\phi \phi(X) > \theta$ that can decide whether or not "the number of points in X is odd" unless at least one of the ϕ's depends on all the points of R.

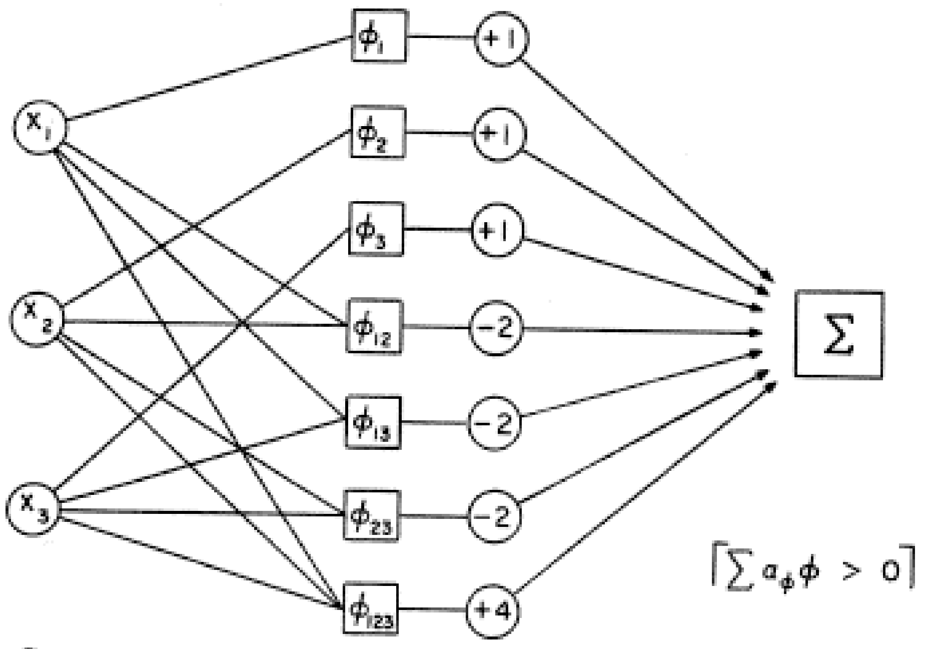
Thus no bound can be placed on the orders of perceptrons that compute this predicate for arbitrarily large retinas. To realize it, a perceptron must start with at least one ϕ that looks at the whole picture! The proof uses several steps.

Step 1: In §1.1 – §1.4, we define "perceptron", "order", etc., more precisely, and show that certain details of the definitions can be changed without serious effects.

Step 2: In §1.3 we define the particularly simple ϕ functions called "masks". For each subset A of the retina, define the mask $\phi_A(X)$ to have value 1 if the figure X contains or "covers" all of A, value 0 otherwise. Then we prove the simple but important theorem (§1.5) that if a predicate has order $\geq$ K (see §1.3) in _any_ set of ϕ functions, then there is an equivalent perceptron that uses only _masks_ of size $\geq$ K (see §0.2).

Step 3: To get at the parity – the "odd-even" property – we ask: What _re-arrangements of the input space R leaves it unaffected_? That is, we ask about the group of transformations of the figure that have no effect on the property. This might seem to be an exotic way to approach the problem, but since it seems necessary for the more difficult problems we attack later, it is good first to get used to it in this simple situation. In this case, the group is the whole permutation group on R – the set of _all_ re-arrangements of its points.

Step 4: In §2 we show how to use this group to reduce the perceptron to a simple form. _The group-invariance theorem_ proved in §2.2 is used to show that, for the parity perceptron, _all masks with the same support size – that is, all those that look at the same number of points – can be given identical coefficients_. Let β_j be the weight assigned to all masks that have support size = j.



Group-invariant coefficients for |R| = 3 parity predicate.

Step 5: It is then shown (in §3.1) that the parity perceptron can be written in the form

$$\sum_{0}^{k} \beta_j \binom{|X|}{j} > 0_1$$

where $|X|$ is the number of points in X, k is the largest support size, and $\binom{|X|}{j}$ is the number of subsets of X that have j elements.

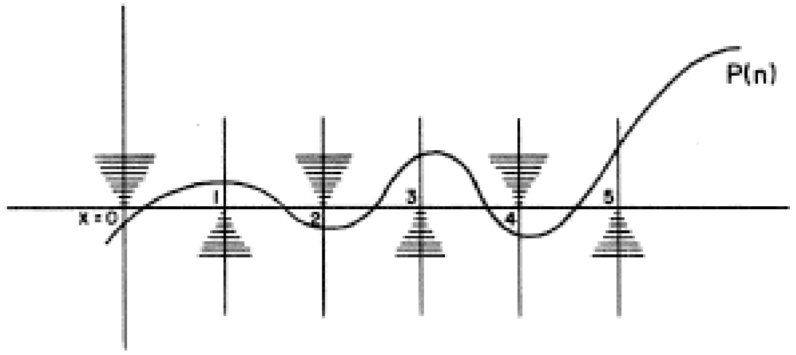
Step 6: Because

$$\binom{n}{j} = \frac{1}{j!}(n)(n-1)\ldots(n-j+1)$$

is a product of j linear terms, it is a polynomial of degree j in n. Therefore we can write our predicate in the form

$$P_k(|X|) > 0,$$

where P_k is a polynomial in $|X|$ of algebraic degree $\leq k$. Now if $|X|$ is an odd number, $P_k(|X|) > 0$, while if $|X|$ is even, $P_k(|X|) < 0$. Therefore, in the range $0 \leq |X| \leq |R|$, P_k must change its direction $|R| - 1$ times. But a polynomial must have degree $\geq |R|$ to do that, so we conclude that $k \geq |R|$. This completes the proof.



This shows how the algebra works into our theory. For some of the more difficult theorems we need somewhat more algebra and group theory.

Here are some of the positive results: that certain patterns have certain orders.

Patterns of Order 1

§0.8 "The center of gravity lies to the left of a certain given point on the X-axis."

§2.4 Other similarly defined properties of moments, in fixed coordinate systems. Includes "The area of the image is less than A."

§1.5 Linear threshold inequalities.

§1.4 "The image is <u>exactly</u> a certain one", or "differs from it by not more than a given area A."

<u>Patterns of Order 2</u>

§7.3 "The figure is symmetrical about a fixed point in the plane."

§7.9 "Two figures, on two given lines, are congruent under translation." (The coefficients diverge, however, as the retina size grows.)

§1.6 "The area of the image lies in a certain range."

§6.2 "The figure lies within an axis-parallel line."

§0.8 "The moment of inertia of the figure exceeds some threshold."

<u>Patterns of Order 3</u>

§7.10 "Two figures on two given lines are translation-equivalent with bounded coefficients w_i."

§6.3 "The image is a <u>convex</u> figure."

§6.3 "The image is an axis-parallel rectangle."

§7.7 "The image is a square." (Coefficients diverge)

§6.3 Any two instances of figures not translation-equivalent can be distinguished.

<u>Patterns of Order 4</u>

§7.7 "The image is a square parallel to the axis." (Bounded coefficients)

§6.4 "The image is a circle."

§7.5 "The image has a vertical axis of symmetry." (Coefficients diverge) (Believed to be unrecognizable in <u>any</u> order for bounded coefficients)

§7.4 Reflection symmetry on a line.

§5.8 "The Euler Number – that is, the number of Components minus the number of Holes – exceeds a given number.

<u>Patterns of Order 5 (or possibly one less)</u>

§8.3 "There is a certain number of convex objects." (No holes permitted)

§8.3 "The <u>total</u> curvature of all the boundaries lies in a certain interval."

§7.5 "Two plane figures are equivalent under translation." (Unbounded coefficients)

<u>Patterns of Order 6 (probably)</u>

§7.9 "Two plane figures are equivalent under translation and dilation."

The remarkable thing about the above results is that the order remains fixed, regardless of the size of the retina. But for other patterns, the order increases without bound as the retina is made larger, and it is fair to say that these are, in a practical sense, outside the conceptual ability of perceptron-like machines. These include:

Patterns of Unbounded Order

§3.1 "The number of occupied retinal cells is odd." (The order is known to be equal to the number of points in the retina.)

§5 "The image is a single connected whole." (The order is known to grow faster than $\sqrt{N}$, probably grows as $\frac{1}{2}N$.)

§5.8 All topological properties except simple functions of the Euler Number. For example: "one part of the image lies within a hole inside another part" cannot be recognized.

§4.0 Certain simple Boolean combinations of pairs of first-order patterns. This and §3.1 show strikingly how perceptrons differ from serial computers, for these patterns are very easy for serial machines.

§6.6 Very few of the finite-order properties mentioned above remain finite order in the context of other figures, or of noise.

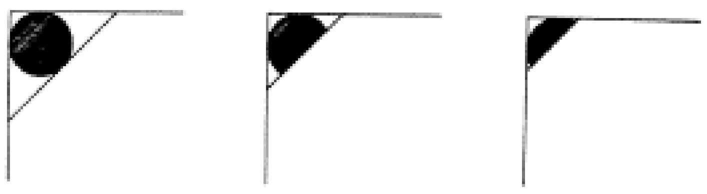
C. Connectedness and Serial Computation

Our deepest results in the perceptron theory are concerned with the geometric – or, rather, topological – properties of connectedness. Because we felt that there was some inherently serial – or recursive – character to connectedness, we decided to investigate the computational geometry of this concept in the context of other mixtures of serial and parallel machine structures, including Turing Machines and Iterative Arrays. Some of these are discussed in Chapter 9 of Perceptrons. We can give here an example of one such result.

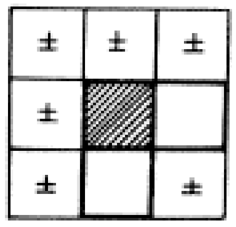
Terry Beyer has investigated the time necessary to compute $\psi_{\text{connected}}$ in a situation that provides a different and perhaps more natural model for parallel geometric procedures. Suppose that each square of a retina contains an automaton able to communicate only with its four neighbors. It can also tell the state (black or white) of its square. The final decision about whether the figure is connected or not is to be made by some fixed automaton, say the one in the top left-hand corner. On the assumption that the states change only at fixed intervals of time, we ask how many time units must pass before the decision can be made. It is obvious that on an $n \times n$ retina this will take at least 2n time units, for this is the time required for any information to pass from the bottom right corner to the top left. It is not difficult to design arrays of automata that will make the decision in the order of N time units, where N is the area of the retina. Beyer's remarkable result is that $(2+\epsilon)\sqrt{N}$ is sufficient, where ϵ can be made as small as one likes by allowing the automaton to have sufficiently many states.

Thus the order of magnitude of time taken by the array is proportional to $\sqrt{|N|}$, which is intermediate between the times taken by the single serial machine (N) and the most general type of parallel computer which is known to take $\leq$ (log N)^2. (Perceptrons, Chapter 9)

The following gives an intuitive picture of Beyer's algorithmic process. The over-all effect is that of enclosing a component in a triangle as shown below, and slowly sweeping it into the northwest corner by moving the hypotenuse inward.



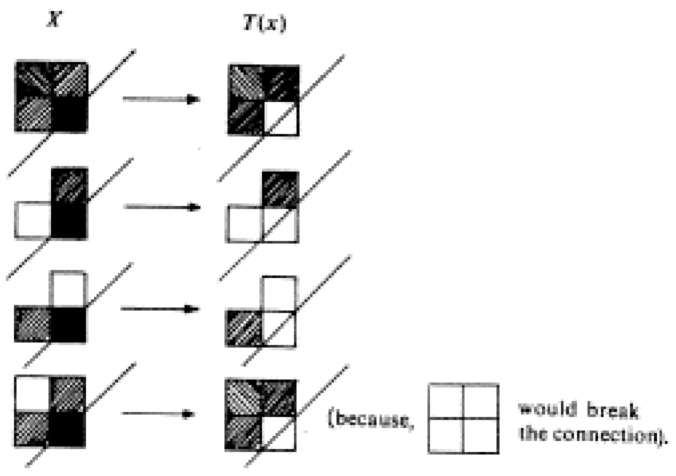
Each component is compressed to one isolated point before vanishing. Whenever this event takes place, it can be recognized locally and the information transmitted through the network to the corner. Thus the connectedness decision is made positively or negatively, depending on whether such an event happens once or more than once. More precisely, the compression process starts by finding the set of all "southeast corners" of the figure.



In this theory the retinal points are taken to be a square checkerboard-like array. Two black squares are connected if they share a common edge, or are in a chain of squares so connected. (Diagonal corner-contact does not count as a connection.)

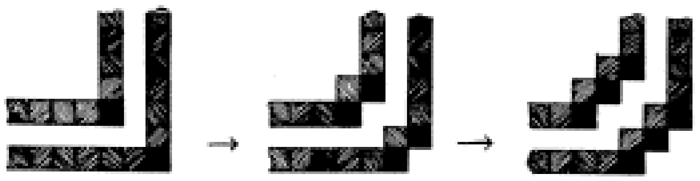
The center square is a southeast corner if the South and East are empty. All other squares shown may be empty or full.

In each step of the compression operation, every southeast corner is removed, and a new square is inserted when necessary to preserve connectedness, as shown below:



The diagonal lines show how repetition of this local process does squeeze the figure to the northwest.

Repeated applications of this operation eventually reduce each component to a single point. The next figure shows how it (narrowly but effectively) avoids merging two components.



It is easy to see that a component within a hole will vanish (and be counted) just in time to allow the surrounding component to collapse down. We do not know any equivalent process for three dimensions. (Consider knots!)

We believe that further investigations of this sort should lead to a deeper understanding of parallel computation in general and of the potential role of such processes in practical artificial and biological visual systems.

D. Designing a Stereo Vision System

Another example of computational geometry is the reconstruction of a three-dimensional scene from stereo views. In very simple cases – for example, where the scene consists of a single point or line segment – the problem belongs to perspective geometry. But scenes such as those shown in discussing SEE raise different problems. One possible approach is to make an analysis using SEE on each stereo view and then match identified objects. Another "pure" approach is to use the stereo match as a means of object identification. In practice, a real vision system would try to use a mixture of both. In this case a new kind of difficulty arises: that of manipulating data of varied kinds and varied degrees of reliability.

David Perkins has been developing for his Ph.D. thesis a programming system for handling complex net-like data structures that could allow information of diverse sorts to be built up in the course of exploring a scene. The stereo problem is being used as the subject-matter for the system.

E. Theorem-Proving Programs

We are concerned about adapting automatic heuristic deductive programs to problems with large and diverse data bases. Gerald Sussman has programmed a classical theorem-prover based on the resolution principle and has begun to gain experience with the introduction of heuristics of a more specific sort than are usually employed. Carl Hewitt, at the other extreme, has developed a system and language called PLANNER to permit the most diverse possible operations that might be used in proofs. Neither project is yet sufficiently advanced to warrant detailed analysis of results.

Chess and Game-Trees – Richard Greenblatt

During the year, substantial progress was made on Mechanical Chess – a domain that has long been of great interest to workers in Artificial Intelligence. The program, known as MACHAC VI, has achieved the level of ability of an average amateur player and has been accepted as an honorary member of the American Chess Federation. The clear superiority of this program over all previous ones is interesting both theoretically and as a visible demonstration of programming techniques and systems (See A.I. Memo 174).

For a long time we have hoped to see a clarification of the theory of non-terminal Evaluation functions in game-playing, tree-search programs. One's first inclination is to try to interpret the E-function as an approximation to an estimate of probability of success, improved by the look-ahead procedure. No one has been particularly successful at making this interpretation workable. Some promising results on probabilistic strategy theory are given in a recent thesis by David Johnson.

Mathematical Laboratory – William A. Martin and Joel Moses

It has been our long-term goal to develop a man-machine system for amplifying the effectiveness of a mathematician working on either applied or abstract theoretical problems. This project is proceeding in several directions, to bring together such systems as the previously reported Integration system of J. Moses, the Symbolic Mathematical Laboratory of W. Martin, and the MATHLAB of C. Engelman. A two or three-year project is planned, leading to an on-line system to aid in carrying out complex and tedious symbolic calculations, and to build up an active store of knowledge about mathematical algorithms and heuristic methods. We are preparing a monograph on symbolic algebraic manipulations. Martin is pursuing the theory of parsing context-free expressions, both as pure theory and in connection with an on-line parser for hand-written mathematical formulae. Moses is developing a simplification system of advanced power that will, for example, be able to convert

$$\frac{1 + 2\log(\sin^2 2y + x + \cos^2 2y) + \log^2(1 + x)}{\log(1 + x) + 1}$$

to

$$1 + \log(1 + x) \quad .$$

With this, it should be feasible to implement a powerful decision procedure for integration due to Risch; the system should be able to integrate

$$\frac{\log z - 2}{(\log^2 z + z)^2} + \frac{\frac{2}{z}\log z + \frac{1}{z} + 1}{\log^2 z + z}$$

to

$$\frac{-\log z}{\log^2 z + z} + \log(\log^2 z + z) \quad .$$

A program, called SARGE, which drills students in freshman calculus integration problems has also been written by Moses. Students are supposed to type step-by-step solutions to an integration problem. The computer checks the correctness of the justification (e.g., substitution of variables) given for each step. Sometimes, though rarely, the computer can give a reasonable description of the type of mistake the student made in an erroneous step.

Interactive Computer-Mediated Animation - Ronald M. Baecker

Standard techniques of man-machine graphical communication have been supplemented with the capability for immediate viewing of a synthesized animation sequence. These were constructed using three special-purpose animation systems which have been implemented with the interactive graphics capability of the Lincoln Laboratory TX-2 computer.

The concept of a table-driven algorithmic synthesis of an animated display has been developed as an approach to the specification of picture dynamics. The tables, called movement descriptions, abstract aspects of behavior that recur over extended intervals of time in a particular animated display. Rhythm descriptions express patterns of the triggering, pacing, coordination, and synchronizing of picture change.

Techniques of control that particularly exploit the interactive environment have been developed. The animator is coupled to the film under construction, both through the sketching and graphical editing of static pictures of dynamic behavior (expressed by movement descriptions), and through the dynamic mimicking of desired behavior. The animator's dynamics are expressed through stylus, push-buttons, and other devices; the language may be used to construct animated visual displays, to build system tools that aid the construction process, and to implement special-purpose animation systems.

Computer time and support for this research have also been provided by the M.I.T. Lincoln Laboratory, under a contract from the Advanced Research Projects Agency.

Fourier Transform Methods in Image Processing - Berthold K. P. Horn

The two-dimensional Fourier transform has been used widely in optics for the evaluation of image-forming systems, yet has been used infrequently in pattern recognition and perception studies. It was thus of interest to find areas of image processing to which the Fourier transform could profitably be applied. Although the Fourier transform has many useful properties that indicate its use in image processing (such as translational invariance) many of these are shared by other functions.

The Fourier transform was found to be useful either in spectral analysis of small image areas, or as a global image-transformer in:

1) focusing an optical device,
2) restoring a degraded picture,
3) resolution beyond the usual limits,
4) edge detection in a preprocessing pass,
5) confirmation of edges in a given position,
6) least-squares filtering in the presence of noise,
7) dynamic range reduction.

No conclusions have been reached about the usefulness of Fourier methods in texture description or curvature detection. A computer program was developed for the PDP-6 to allow experimentation on images read in through the on-line computer "eye", and the Fast Fourier Transform method was used in this program.

Structure of Atonal Music - Allen Forte

Some aspects of this project, which is concerned with a general structural description of so-called atonal music, have been presented in MAC Progress Report III and, more extensively, in MAC-TR-39. During the past year the score-reading program, which parses an input string representing standard music notation, was improved in the direction of greater generality and efficiency, and more effective tools were developed to deal with certain basic problems of musical analysis. In particular, the notion of time-point states has led to a solution to the problem of segmentation (the determination of structural units) and has suggested some useful ways in which relations between such units might be displayed. The task of constructing an appropriate model for this music remains difficult. Work done thus far has suggested several possibilities which are currently being programmed.