WORKING PAPER 63

AN HYPOTHESIS-DRIVEN RECOGNITION SYSTEM FOR THE BLOCKS WORLD

BENJAMIN J. KUIPERS

Massachusetts Institute of Technology

Artificial Intelligence Laboratory

March 1974

Abstract

This paper presents a visual recognition program in which the recognition process is driven by hypotheses about the object being recognized. The hypothesis suggests which features to examine next, refines its predictions based on observed information, and selects a new hypothesis when observations contradict its predictions. After presenting the program, the paper identifies and discusses a number of theoretical issues raised by this work.

Work reported herein was conducted at the Artificial Intelligence Laboratory, a Massachusetts Institute of Technology research program supported in part by the Advanced Research Projects Agency of the Department of Defense and monitored by the Office of Naval Research under Contract Number N00014-70-A-0362-0005.

An Hypothesis-Driven Recognition System for the Blocks World

Benjamin J. Kuipers

This paper presents a program for recognizing line drawings of several simple kinds of blocks. Its purpose is not primarily to make another attack on the visual problems of Blocks World, but to explore some of the issues of control flow that have come up in connection with various proposals about Frames.

> A frame, in this context, is a detailed hypothesis about the structure of the line-drawing being explored. A collection of frames is linked by a transition network. The frames span the domain of objects we are interested in recognizing, and the difference net uses anomalies (observations inconsistent with the current hypothesis) to select a better frame to direct the recognition process.

This program looks at a line-drawing of a single, unoccluded block, and attempts to classify it either as a parallelopiped (with three visible faces), or a wedge, which may, in general position, have either two or three visible faces. (See figure 1.) The input is a simulated vidissector (described in more detail below) which gives very local information about the scene. In order to preserve the qualitative nature of this investigation, only the very crudest metrical data is available to the program.

The program I describe was initially written to explore some of the ideas presented by Minsky in his theory of Frame-Systems [Minsky 1974]. It fits into the context of Blocks World vision, which has been extensively explored by Winston, Waltz, and others [Winston 1973]. I make use of many of their ideas, particularly the difference network

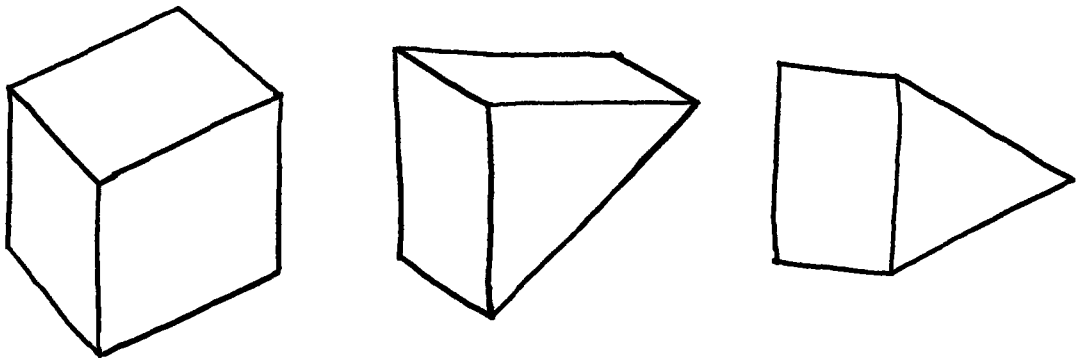


Figure 1 The domain

which was proposed by Winston.

There is an important contrast between my block-frame and Waltz' program. His program knows a great deal about individual vertices, and imposes a consistency restriction on the edge connecting two vertices. The local, vertex-based knowledge propagates around the network via the consistency requirements on the edges, and allows the system to come to a global conclusion about the nature of the scene. This is basically a local strategy, in the sense that the program has only local knowledge. That it works shows the great extent to which the objects in our perceptual world are determined by local consistency. The block-frame, on the other hand, is a detailed global hypothesis which is much more willing to use a local fact, along with its own assumptions, to reach a conclusion about a remote part of the scene. The obvious advantage is that observations tend to be verifications rather than discoveries. The disadvantage is that it also tends to be wrong more often. This,

surprisingly, is not so bad if the system has the ability to detect an error and select a new hypothesis based on the nature of the error, and to retain most of the information collected under the old (false) hypothesis.

More similar to this work is a paper by Shirai [in Winston 1973], who directs the image analysis through consideration of knowledge about the kinds of objects to be expected in the scene. He, like Waltz, deals with local heuristics which guide the search for edges near a particular vertex, and vertices along a particular edge. His work presents much of the justification for the design of my simulated vidissector. Based on Shirai's work, Lerman and Woodham [1973] have produced a collection of LISP functions to manipulate vidissector data for vision work. Their primitives would provide a natural way to implement the vidissector on real data.

There are also some similarities between the hypothesis-driven recognition process and template-matching. One distinction is that the possible variation in a template is typically a range of values for each of a small number of numerical parameters, while the frame can specify the range of variation with an arbitrarily smart procedure. Another important difference is that violations of the permissible range point to other, particular, frames rather than simply refuting the current one. And finally, the knowledge that can be globally communicated to remote parts of the frame may be arbitrarily complex.

A Simulated Vidissector

The aim of this research is to investigate the control structure needed to manage an hypothesis-driven recognition system. To be able to concentrate on these questions rather than the image analysis problems involved in using real vidissector output, I have designed a simulated vidissector. This device provides several natural primitives for exploring simple line drawings. The drawing is represented as a collection of edges and vertices, each of which may be asked for certain information about it and its immediate neighbors. The primitive operations are the following:

> A vertex will deliver the edges which terminate at it, and the sizes of the angles between pairs of edges. There are three types of vertices: L, Y, and arrow. An angle measurement may take one of three values: acute, right, or obtuse. This corresponds to the result of a "circular search" in the neighborhood of a vertex.
>
> An edge will deliver its "other vertex" upon being presented with one vertex. This corresponds to following an edge from one vertex to another.
>
> Two vertices may be asked if they are "the same". This corresponds to comparing the absolute coordinates of the points in the scene.

How does this simulation allow the theory to extend to richer, more quantitative data? There are two points to make here. The first is that humans are able to recognize faces and other familiar objects from very crude, low-resolution data. An adequate theory of recognition cannot be dependent on high quality data. Second, and more important, is that excess data is a major problem to vision research, in the form of spurious lines and features which are picked up in the attempt of find low-intensity features. An hypothesis-driven recognition system, by

predicting where features should appear in the scene, allows the recognizer to verify, rather than discover, most of the scene. Thus many spurious features will not be processed at all, since they do not appear in a region of interest to the hypothesis, and others can be dealt with by the complaint department mechanism which handles anomalies. In order to concentrate on the hypothesis, I have deliberately side-stepped several of these important issues.

As an example, consider the image of a cube (figure 2). The

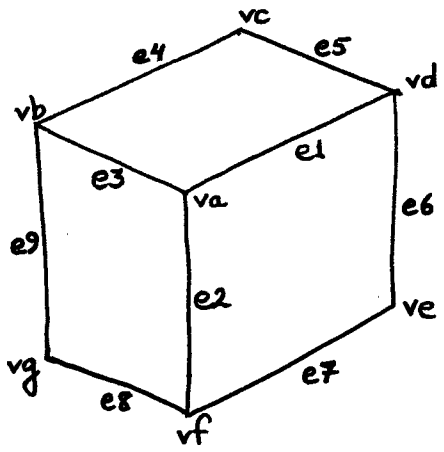


Figure 2 The simulated vidissector image of a cube

edges are represented by procedures which will answer questions about the "other vertex". These are set up by a master procedure called 'edge'. For example, if el [which was produced by the function call (edge va vd)] is sent the message "other vd", it will reply "va". When the recognizer has received a vertex, it has access to any of the edges terminating at that vertex, and the sizes of the angles between those edges.

Below are presented the data which make up the simulated

vidissector image of figure 2.

The Vertices:

```
va:     (#Y-vertex (edges: e1 e2 e3) (angles: obtuse obtuse obtuse))
vb:     (#arrow (edges: e9 e3 e4) (angles: obtuse acute obtuse)
vc:     (#L-vertex e4 e5 obtuse)
vd:     (#arrow (edges: e5 e1 e6) (angles: acute acute obtuse))
ve:     (#L-vertex e6 e7 obtuse)
vf:     (#arrow (edges: e7 e2 e8) (angles: acute obtuse obtuse))
vg:     (#L-vertex e8 e9 obtuse)
```

The Edges:

```
e1:     (edge va vd)
e2:     (edge va vf)
e3:     (edge va vb)
e4:     (edge vb vc)
e5:     (edge vc vd)
e6:     (edge vd ve)
e7:     (edge ve vf)
e8:     (edge vf vg)
e9:     (edge vg vb)
```

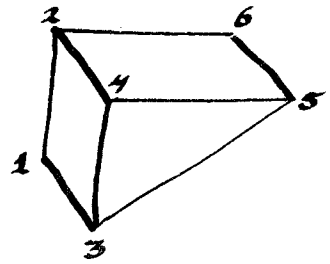
```
(edge <=
  (=> [=v1 =v2 =L]
    (cases
      ((=> [#other =v]
         (rules v
           ((=> v1 v2)
            (=> v2 v1))))
       (=> [#length] L)))))
```

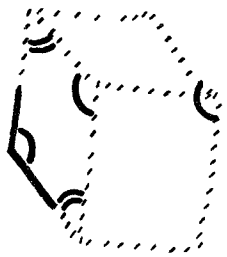
The recognition system is coded in an experimental formalism called ACTORS, which was developed by Hewitt as the latest in the PLANNER family of languages. [see Hewitt 1973] In ACTORS, a program is considered as a collection of modular agents (the actors), and the primitive (and only) operation is sending a message to an actor. This view of computation has many controversial implications which do not concern us here. I have used ACTORS for the recognition program because

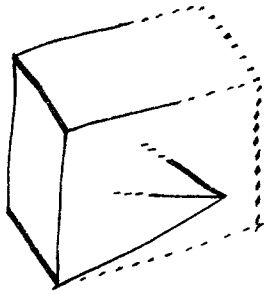
it allows me to manipulate the flow of control and information much more freely than a more functionally-oriented language, such as LISP.

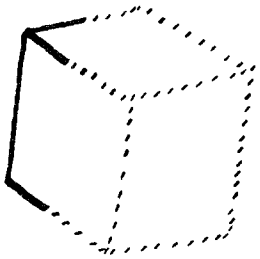
The syntax of the edges and vertices should be fairly clear. The syntax of the procedure "edge" requires a little explanation. Note that (=> pattern body) means "Wait for a message that matches pattern. When you receive it, execute body." The symbol "=>" is read "receives". If body returns a value, that value is sent as a message to the continuation supplied by the caller; i.e. that value is returned as the value of the receives statement. We may now read the procedure as prose, including explanations of the cases and rules statements.

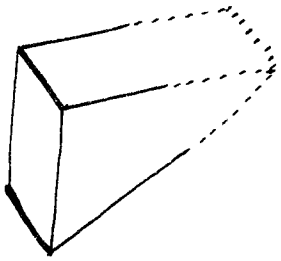
> The actor named "edge" waits for a message which is a 3-tuple. When it receives that message, it binds the values of the elements to the names v1, v2, and L. It returns the cases statement with these bindings. The cases statement also waits for messages. Upon receiving a message, it sends the message to each of its clauses. If none of the clauses will match it, the cases statement returns "Not Applicable". To match the first clause, the message must be of the form "other v". The rules statement then attempts to match v against each of its clauses. In this case, if v matches v1, v2 is returned, and vice versa. If v matches none of the clauses of the rules statement, "Not Applicable" is returned. If, on the other hand, the message to the cases statement is "length", then the value of L is returned.

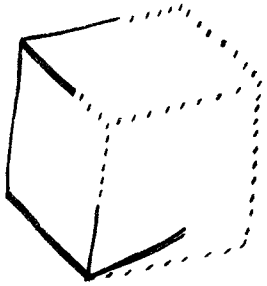


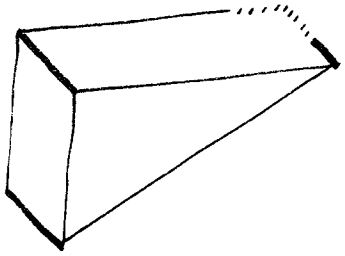


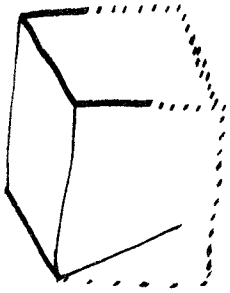












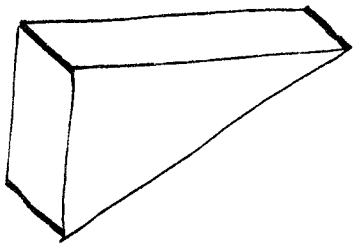


Figure 3 Recognition scenario

A Recognition Scenario

Let us follow a scenario of the recognition of a block drawing, in this case of the three-face view of a wedge. Figure 3 shows the stages of the recognition process, with observed data indicated in solid lines and hypothetical knowledge in dotted lines. The first figure is the actual scene, with the vertices numbered in the order in which they will be explored.

Vertex 1

We must start the recognition process by giving the program an initial vertex, which in this case happens to be an L-vertex. Our initial hypothesis is that the figure is a parallelepiped, indicated by the dotted lines in the figure. The single angle-measurement has provided expectations for the four additional angles indicated.

Vertex 2

The second vertex observed fits in completely with the hypothesis, which expected an arrow vertex, and had a particular measurement anticipated for the right side angle of the arrow. The two other angle measurements provided by the arrow, added to the angle database, complete the specification of expected values for all of the angles in the figure, using knowledge of the global angle relations in the parallelepiped (figure 4).

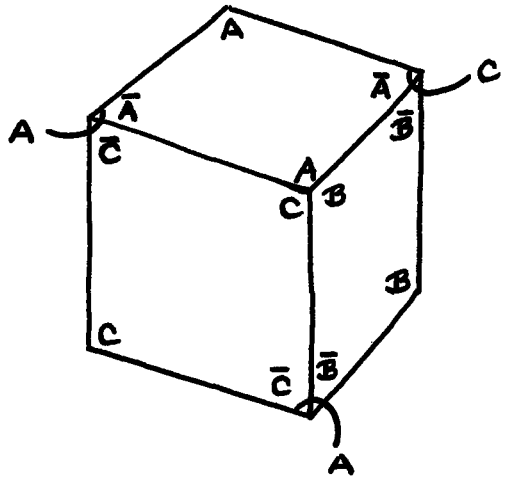


Figure 4 Global angle relations in the parallelepiped frame

Vertex 3

This vertex fits the hypothesis of the vertex-specialist, since it is the anticipated arrow vertex. At this point we can see that the angle is too small, and that the figure cannot be a parallelepiped. If I had endowed the program with better angle-resolution, the angle database would also notice this and complain to the frame. However, it cannot discriminate well enough, so the angle database accepts the information as consistent, and the recognizer continues with a mistaken hypothesis. We will see later how it recovers.

Vertex 4

The Y-vertex at the center of the figure also corresponds completely with the parallelepiped hypothesis. A complete parallelogram face has now been observed and confirmed.

Vertex 5

With this observation, the parallelepiped hypothesis finally breaks down. The L-vertex specialist observes an unexpected type of vertex and complains to the frame: "I expected an L, but got an arrow!" The parallelepiped frame knows that this particular problem indicates a transition to the three-face wedge frame, which it orders.

Notice some fancy stepping here, though. The unexpected arrow vertex was an anomaly to the parallelepiped frame, and the information contained in it could not be completely processed. So it was ignored, and the transition to the wedge-frame took place with only the previously-known data. Once the new frame was in control, it could deal with the arrow vertex. The arrow vertex, in effect, caused the recognition system to do a "double-take".

Vertex 6

At this point, with the three-face wedge frame directing the exploration, there is only one remaining vertex, and it completely confirms this hypothesis. The frame is now fully instantiated.

The Recognition Program

In this section, we will see the details of how the block recognition program works: what procedures it consists of, what strategies and knowledge it contains, how it represents its hypothesis, and how it copes with anomalies. The next section, entitled What Have We Learned? will cover the same topics from a more general and theoretical point of view.

What Procedures Does It Consist Of?

The recognition program as a whole consists of three frames and the transition net. The three frames are for a parallelepiped, the two-face view of a wedge, and the three-face view of a wedge. The three frames are similar in structure, though the parallelepiped frame is the most elaborate because of its central position in the transition net. (See figure 5.) We will focus on the parallelepiped frame, then, since it represents more of the behavior we are examining. A frame consists of a number of component specialists, which embody the frame hypothesis, and a complaint department which handles violated expectations.

The component specialists of the parallelepiped frame are:

- 3 parallelogram specialists
- 7 vertex specialists
  - 3 arrow-vertex specialists
  - 3 L-vertex specialists
  - 1 Y-vertex specialist
- 9 edge specialists
- 15 angle specialists
- 1 angle database with associated demons

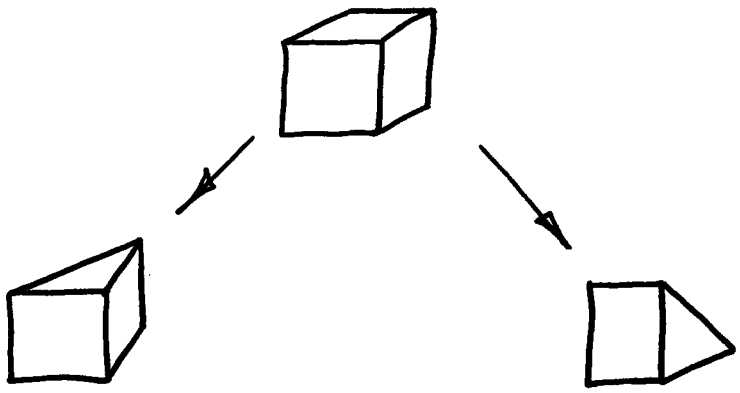


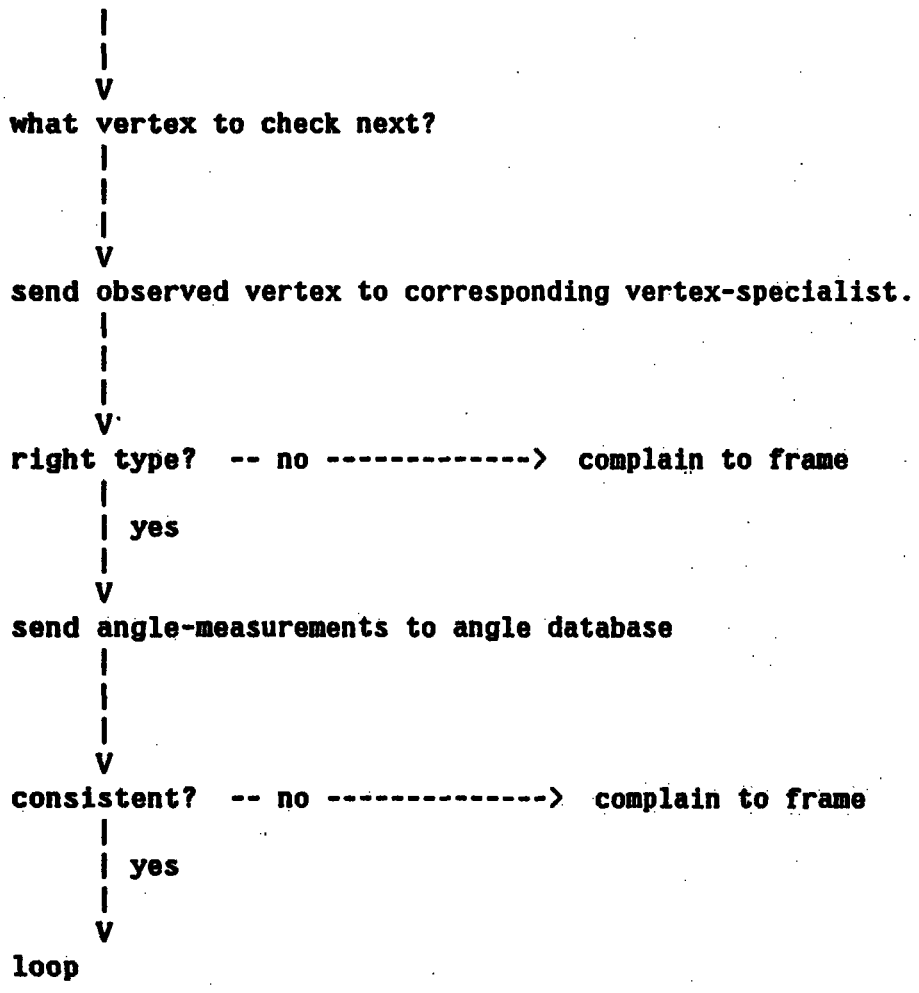
Figure 5 The transition network

The specialists perform several jobs, each of them quite simple. The first is to maintain orientation information: each specialist has pointers to its immediate neighbors and constituents. This orientation information represents the topological connectivity of the object by describing all of the local connections between neighboring features. Each vertex specialist knows which type of vertex it expects to correspond to, and will complain to the frame if it receives a different type. The angle database embodies (in its demons) global knowledge of the angle relations which hold throughout the figure, and deduces what it can from angle-measurements which are asserted to it. It will also complain to the frame if it observes an inconsistency.

The complaint department of the frame receives any complaints from the component specialists about anomalies they observe. It then diagnoses the underlying cause of the problem, and orders the transition net to take the appropriate action.

The Basic Loop

The figure below summarizes the recognition strategy while a particular frame is in control. I have left out the entry to the loop with the first vertex, and the exit when there is nothing left to do.



As long as the observed data continues to confirm the current hypothesis, control cycles through this loop. Very quickly most of the observations become verifications of predicted features.

The Complaint Department

The complaint department in the frame receives complaints from the component specialists, and from its more global point of view, decides what is the underlying cause of the problem. It then orders the appropriate transition. In this small block domain its job is relatively easy since, with one exception, each anomaly unambiguously specifies a transition to a new frame. The one exception (figure 6) activates an exploratory routine to settle the ambiguity.

Complaints from vertices:
expected L, got arrow ----------> to wedge-3
expected arrow, got L ----------> to wedge-2
expected Y, got arrow ----------> to wedge-2 (ambiguous)

Complaints from angle database:
L angle too small ----------> to wedge-2
arrow side-angle too small ----------> to wedge-3
arrow full-angle too small ----------> to wedge-3

Any other complaint means that the drawing is of an object not in the domain, and is an error.

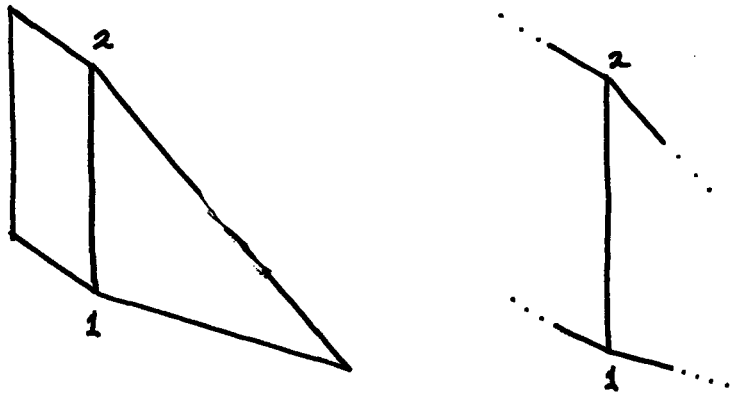


Figure 6 The ambiguous transition
Within the resolution of the angle measurement, the frame cannot predict which side is the triangle.

The complaint department is clearly the critical part of the recognition scheme, since it handles the weak point of an hypothesis-driven method: what to do when the predictions are wrong. In general, the complaint department could deduce the underlying cause of an anomaly from an abstract description of the objects in the domain, given a large amount of deductive machinery. In our case, however, the knowledge in the complaint department is highly compiled from an analysis of the objects in the domain. The restricted domain is largely responsible for the unambiguous diagnosis possible here. For a somewhat more elaborate complaint department with similar purpose, see Sussman [1973].

The Ambiguous Transition

In the case shown in figure 6, where an anticipated Y-vertex turned out to be an arrow, the parallelepiped frame knows that the new frame must be wedge-2, but doesn't know which orientation to assign to it. To solve this, an exploratory routine looks at one of the faces, trying to decide whether that face is a triangle or a parallelogram. When it has decided, this information determines the orientation of the observed data as it is transferred to the wedge-2 frame.

The Transition Net

Once the frame's complaint department has decided which new frame to change to, and specified an orientation for the data, the transition

net must carry out the transformation. The difficulty is in translating the orientation information that applies to the whole figure to specific instructions for dealing with individual pieces of data. For each frame transition, there is a separate problem for transferring data collected by the vertex specialists to the corresponding vertices in the new frame, and for changing the angle database so that the expectations for the new frame are computed.

In both transitions, the predictions in the angle database from the old frame are simply discarded. As the correspondence is established with the parts of the new frame, the observational data will be relabelled, new demons added to the database, and new predictions computed. When changing to wedge-2, the edge joining the two faces makes a starting point from which to step around each face, transferring data from the parallelepiped to corresponding parts of the wedge-2 frame. When changing to wedge-3, one of the parallelograms making up the parallelepiped is instructed to change itself into a triangle. It does this, informs all of its neighbors of the change, passes the correspondence information on to the angle database, and the transition is complete.

There is further discussion of these transitions in a following section entitled What Have We Learned?

What To Do Next?

The first vertex is simply given to the recognition system to

start the whole process. The algorithm to decide which vertex to investigate next is simple to describe, though annoyingly complicated to implement:

> A vertex is known if the data corresponding to it has been observed and stored.
>
> A vertex is accessible if one of its neighbors is known.
>
> The frame selects the face which contains the most known vertices without being complete.
>
> That face selects the first of its vertices which is accessible but not known. The parallelogram orders its vertices: outer, right side, left side, center. (see figure 7.)

As we shall see below, there are other search strategies, and reasons for choosing them.

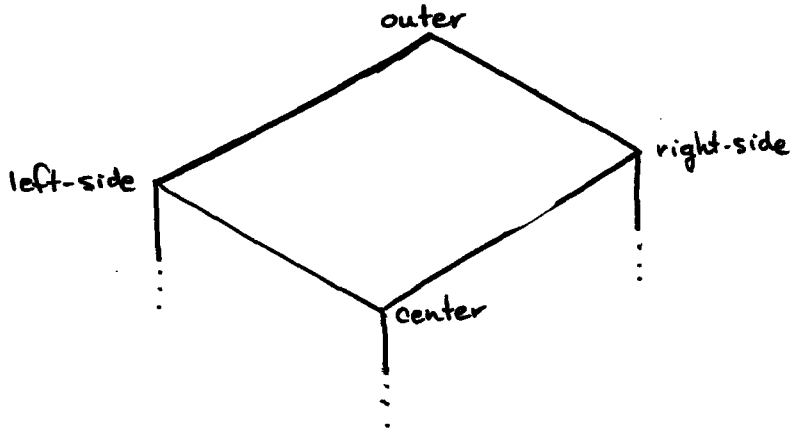


Figure 7 How the parallelogram specialist sees its vertices

What Have We Learned?

At this point, it is appropriate to reflect on what we have learned from this experience. I have presented a program which employs a large amount of machinery to recognize a very simple scene, of a kind that has been thoroughly explored by other workers. My goal in choosing this problem has been to use a simple, well-understood domain to explore some of the characteristics of an hypothesis-driven recognition system. Further research will focus on the problems that arise in a less trivial domain. I would like to reflect now on the structure of the program, the design decisions I made, and the general issues which this program illustrates. In writing the program, when two alternatives at a decision point seemed attractive, I tried to find a place to implement both of them. We shall see an example of this below.

I am particularly interested in identifying those parts of this recognition scheme which are dependent on the particular domain. One reason for this is to identify design decisions that are dependent on the nature of the domain, to describe the particular features which determined that choice, and to enumerate the alternatives which may be useful in other domains. Another reason is to point out characteristics of my recognition program which I have included to explore some idea, but which may appear unnecessary or unimportant without special mention.

Correspondences

When using a global hypothesis to guide the investigation of local data, the first problem that arises is establishing a correspondence between the data and the hypothesis. The classic example of this is the story of the six blind men and the elephant. In my limited domain of blocks the problem does not arise immediately. The drawing of a parallelepiped is symmetrical, so the first vertex found can be assigned to any vertex-specialist of its own type (L, Y, or arrow) without risk of later conflict. Even if we add a distinguished direction ("down") to the scene, the choice of correspondence can still be made unambiguously.

A problem does arise when an anomaly forces a transition to one of the wedge-frames, which are not symmetrical. In most cases, the frame can decide how the previously collected data corresponds to the parts of the new hypothesis. However, there is one case (figure 6) when the frame knows that the current hypothesis must be changed to the two-face wedge, but still doesn't know which face will be the triangle and which the parallelogram. There are two alternative courses of action here. One is simply to pick an orientation and switch to the other one if a later anomaly arises. This in effect provides two symmetrical frames corresponding to the two orientations of this view of the wedge. The other choice is to allow the transition net to launch its own investigation which would explore new parts of the drawing until the issue was decided. The best alternative obviously depends on the properties of the particular domain; I have chosen the second

possibility. In fact, however, the order in which the vertices are normally searched prevents this situation from arising.

Domains

What should be the elements of the domain? That is, how many frames do we need in the transition net to describe the objects in the domain effectively? In the best of all possible worlds, an anomaly is immediately apparent and unambiguously selects a transition to another frame. The example given above under Correspondences shows how this can fail: an anomaly appears which considers two distinct frames equally likely. The alternatives proposed there amount to adding another frame to the network, or giving more look-ahead power to the diagnostic procedures that decide what to do about the anomaly.

Under other circumstances than this simple block domain, two other possibilities come to mind. One is that you may discover an anomaly, but can do nothing with it, and so just remember it until the next anomaly appears, when you can consider them both together. This agrees with the philosophy which says that an hypothesis known to be incorrect may be better than no hypothesis at all. The other possibility is observing something which may or may not prove to be an anomaly, depending on data to be collected later. Under these circumstances, Goldstein [1974] generates what he calls caveats, which lend their weight to certain interpretations of later anomalies, but which do not themselves constitute a violation of the hypothesis.

These four situations occur trivially, if at all, in my simple block domain. However, they can occur in domains with a more complex structure. Which of them can occur depends on the terms in which each object can be described, and on what other objects exist in the domain it is compared with.

On Losing Information

Why is an hypothesis-driven recognition system useful? Partially because, based on the assumption that our hypothesis is correct, we can derive global conclusions which would not follow from the data alone. The hypothesis also allows us to select a representation which is optimized for the data we expect to encounter. Part of the price we pay is that an anomalous observation can clash violently enough with the current hypothesis that its symbolic description cannot be represented fully within the current frame. Under these circumstances, the anomalous piece of data is processed just enough to specify the transition to a new frame, and then is sent to the new frame, which has the descriptive machinery to deal with it. Phenomenologically, this corresponds to a "double-take" where you must look again at an object to see features that simply failed to appear while it was incorrectly classified.

In a simple way, this occurs when an L-vertex specialist receives an arrow vertex (figure 3). He can complain to the frame that the L-vertex was missing, which specifies a transition to the three-face wedge frame. However,there is no provision for the large amount of information

carried by an arrow vertex. The alternative of providing the L-vertex specialist with enough extra power to cope with such anomalies is unsatisfactory. Admittedly, this is hardly a serious problem in this context, but it bears on the important general problem of providing sufficient descriptive power for all eventualities. The problem of having to discard or re-evaluate information that is incomprehensible in the current context seems to be an essential characteristic of an hypothesis-driven recognizer. It is the price we must pay for getting conclusions "for free" from the frame hypothesis.

On Databases and Demons

An ubiquitous problem in this kind of programming is: how is the knowledge to be represented? One school of thought would represent everything as procedures which communicate with the rest of the world however they please. An alternate school of thought represents everything as declaratives in a database and deduce all possible conclusions. One of the many compromises between the extreme positions uses procedures known as demons to watch over a database and encourage certain deductions rather than others. A demon is a pattern-invoked procedure with a certain focus of attention, which is activated upon the occurrence of an event matching its pattern. In this context, watching a database of assertions, a demon is essentially a rule of inference of arbitrary complexity, which has two other important properties: it can remove, as well as add, assertions to the database; and it may itself be

activated or deactivated by an external agent, perhaps another demon. It is thus a very powerful and flexible tool for combining the declarative and procedural representations of knowledge.

A particular frame hypothesis is essentially a collection of assertions about the nature of the object being recognized which are simply assumed as part of the recognition process. These assertions constitute knowledge which is used in two quite different ways, and hence represented differently.

First, there is the knowledge about the object being recognized which follows entirely from the frame hypothesis and not from any observational data; for example the topological connectivity of the vertices and edges. This is represented implicitly in the neighbor-pointers that the various specialists maintain. Each of the three faces of the parallelepiped, for example, knows the names of its right and left neighbor faces, its four edges, and its four vertices.

Second, there is the observational data, and the knowledge which is deduced about the figure from it and the frame hypothesis. An example is the angle-measurement which, observed at one L-vertex, specified the size of four other angles. This is represented as an assertion in a database, watched over by special demons which embody the knowledge of the global angle relations. When an angle measurement is added, the demons deduce from it what they can and add their conclusions to the database. They also watch for contradictions, which constitute anomalies challenging the frame hypothesis, and complain to the frame.

It has been suggested that the whole frame could be represented

as assertions in a database. This is, of course, true, but I have found it easier to think about these phenomena if I considered the frame hypothesis to be embodied in procedures, and only the observational data and conclusions drawn from it actually represented as assertions. This impression is very subjective, and could easily change after experience with more complex domains.

Transitions

Once an anomaly has been studied, and the proper new frame selected, how is the transition actually implemented? The answer to this question has poorly understood implications for the procedural structure of a frame. There are two possibilities that have come up, and since they both have advantages, I have implemented them both.

From one point of view, a frame is a procedure which receives data and stores it in a bunch of cells and/or databases. When a transition is indicated, the transition network establishes a correspondence between the cells and databases in the two frames, and simply ships the collected data from one to the other. Some processing may take place in the new frame to extract global deductions from the newly-arrived data. Then the frame goes on its way. This method performs the transition from the paralellepiped to the two-face wedge in my program.

From another point of view, a frame is a collection of sub-frames, which are in turn made up of sub-frames, and each frame has

pointers that allow it to communicate with its neighbors, sub- and super-frames. To accomplish a transition in this model, you simply replace certain of the sub- and super-frames with others, and re-adjust the relevant pointers. This also requires a pass over the database of global conclusions to eliminate falsehoods and encourage the derivation of new truths. I have used this method in the transition from parallelepiped to three-face wedge.

What considerations help decide between these methods? Again we must address the nature of the actual domain. In making the transition to the two-face wedge, most of the features of the frame change: a locally-observed anomaly has global effects on the hypothesis. In changing to the three-face wedge, most of the features stay the same.

This suggests a general question which we can ask of the frame-system, as a collection of hypotheses: when an anomaly appears, an a transition to a new hypothesis is indicated, how much of the old hypothesis is still (to the best of our knowledge) true? Most of the predictions made by the parallelepiped frame are true of a three-face wedge. This is not the case if the figure is actually a two-face wedge. Note, however, that the answer to this question is very dependent on the descriptive mechanism being used. In analyzing line-drawings of these blocks, the two-face and three-face views of a wedge are very different. If we were manipulating three-dimensional models (e.g. for construction) the two models would be virtually identical, while the transition to either from a parallelepiped would have more global effects on the description.

Search Strategy

At each point in the exploration of the figure, the program must decide what it should inspect next, from among the collection of vertices accessible to it. The module which makes this decision is conceptually independent of the rest of the recognizer, in the sense that the various alternatives do not affect whether or not the object will be recognized. What they do influence is the order in which various expectations about the figure are confirmed or denied. Thus alternative search strategies differ according to how much effort must be expended to confirm which features. In substantially more complicated domains, say Chess or Go, the choice of search strategy could conceivably determine whether some configuration was recognized in a reasonable amount of time.

In the small block domain, there are three possible strategies. The first emphasizes verifying a particularly useful set of features, such as a skeleton. (A skeleton of a block is a minimal set of edges that determines the metrical properties of the block.) Notice that, under the assumption that a block is a parallelepiped, you can find its skeleton before discovering that it is, in fact, a wedge (figure 8). The second strategy is to emphasize finding anomalies if there are any. This may delay finding new metrical information in order to know as quickly as possible how to classify the object. The third strategy is to concentrate on completing and verifying sub-frames, verifying the structure hierarchically from the bottom up.

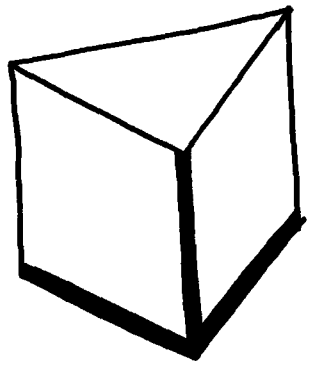


Figure 8 Finding the skeleton of a block before knowing what kind of block it is.

The choice between these alternatives depends on the goals for which the recognizer is being used. As a demonstration program, it is not subordinate to higher goals, so the choice is largely arbitrary. The strategy I use is primarily the third one, with some influences from the second.

And So On ...

There are two ways of thinking about extending this work, depending on whether you regard it as addressing the problems of vision or of representations of knowledge. In the sections below, I try to outline what my program has done, and what needs to be done next in each of the two areas. My own interests lie in the second.

Scene Analysis

Extending this block-recognition scheme from a recognizer of simple unoccluded blocks to general Blocks World scenes will involve the solution of several non-trivial problems. It also will allow the development of a simple domain for exploring a very important theoretical problem: the cooperation of syntactic and semantic analyses of the same phenomenon.

The first set of extension problems deal with occlusion of part of one block by others. A T-vertex is likely to indicate that an edge is occluded by another object. Valuable lower-bound information can still be extracted about the length of that edge, and used to evaluate the consistency of the current hypothesis. The frame must be capable of representing "excuses" for incomplete data in the picture of the block. One of the features of an hypothesis-driven scheme, however, is in extracting maximum information from incomplete data. With a simple method for handling T-vertices, figure 9 could be recognized as one fully

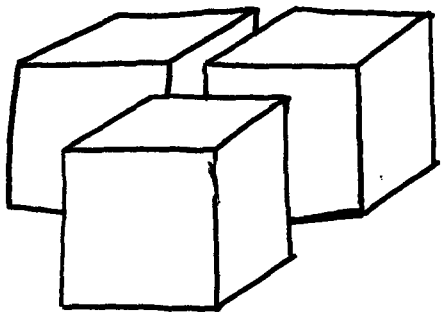


Figure 9 Occluding blocks

instantiated parallelepiped frame and two partially instantiated frames. There are circumstances (figure 10) where parts of the same object are

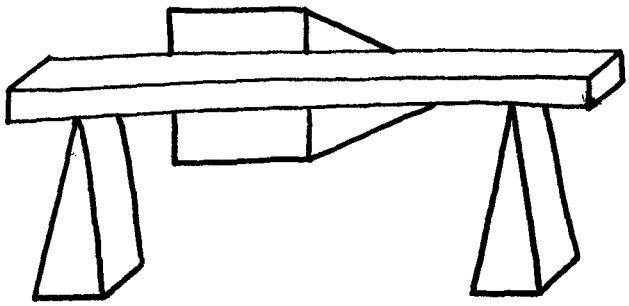


Figure 10 An occlusion which separates a block into disjoint portions

separated by an occluding object, and each part will thus have its own, partially instantiated frame. The techniques must be developed to find a mutually consistent interpretation which will allow the two frames to be

merged. Note in figure 10 that each portion of the occluded block is consistent with a parallelepiped, but together they must be a wedge.

Accidental alignments are a particular problem to scene analysis programs, since they produce apparent vertices which do not correspond to a single part of the scene (figure 11). The hypothesis-driven recognizer

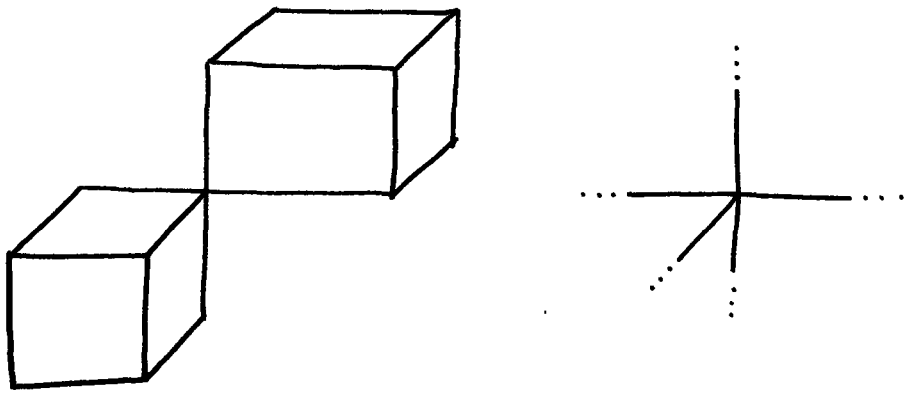


Figure 11 Accidental alignment and the accidental vertex

can deal neatly with these by simply ignoring difficult vertices until it has understood all of the easier ones. Then, it may enter a special "accidental alignment mode" and explain part of a vertex as satisfying its own needs, hoping that other frames will explain the remaining parts. Depending on the control structure, it may even be able to oscillate between interpretations of genuinely ambiguous figures, just as humans do. (See figure 12.)

Shadows represent information global to an entire scene in a way analogous to angles representing information global to a single block drawing. However, identifying a shadow line and extracting its global

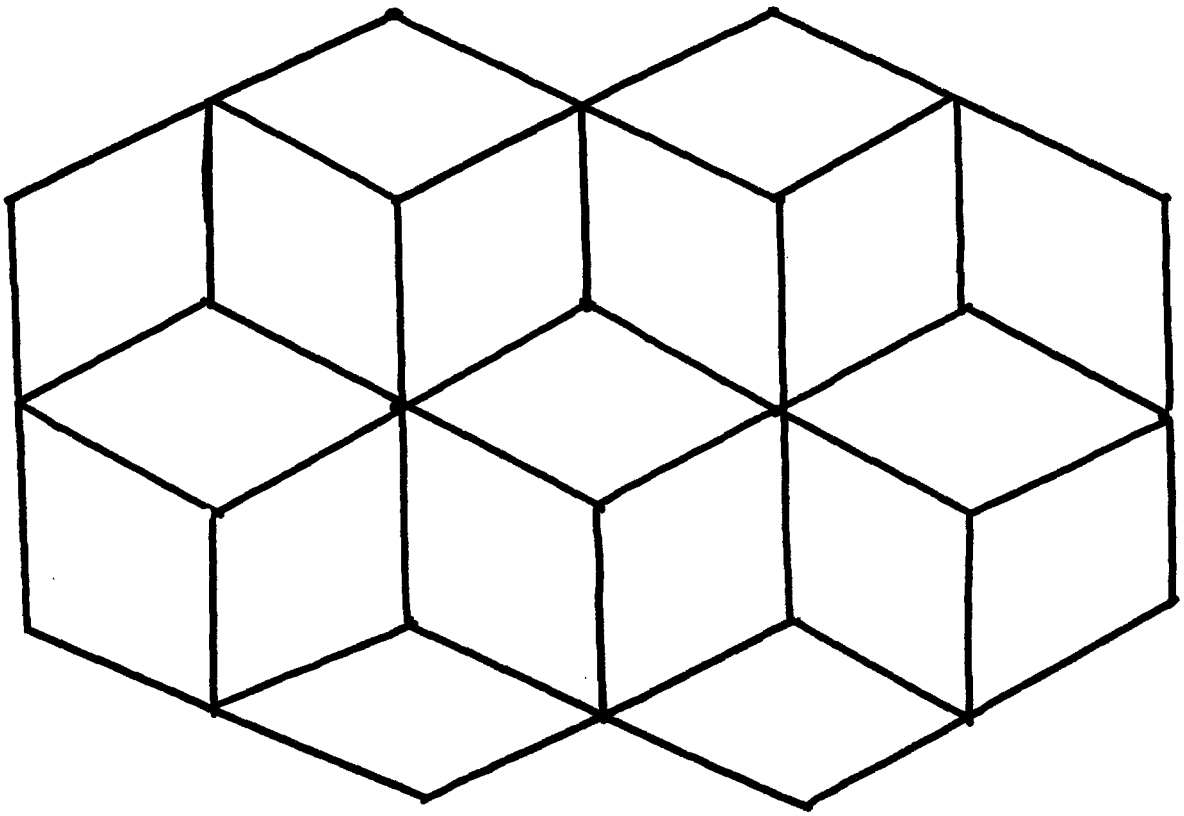


Figure 12 Ambiguous stacked cubes with oscillating interpretation

implications is substantially harder than the corresponding problem for angles.

From one point of view, it could be argued that Waltz' program is a syntactic analysis, since it uses purely local information about possible vertices, while mine is semantic, since it uses a global hypothesis about the nature of the entire structure. This allows us to speculate that these two Blocks World vision programs could let us investigate the problem of cooperation between syntactic and semantic analyses in a relatively accessible domain. The problems in this area

are clearly less difficult than the corresponding ones in natural language, the other field where such cooperation has been proposed.

Things Not Done

The frame structure is used to represent detailed knowledge about a small collection of similar objects. In this case, that knowledge constitutes the hypothesis behind an hypothesis-driven recognition system. Such a representation may also be useful in other applications, such as planning or natural language understanding. There are a number of issues which are of interest in these other applications, and perhaps in more sophisticated recognition applications, which I have avoided in this research.

The part-whole relationship between two objects, and the instance (or KIND-OF or ISA) relationship between an object and a category, are relationships which can communicate properties to an object. In certain domains (e.g. medical diagnosis) there may be no clear anomaly refuting a particular hypothesis, but an aggregate anomaly deriving from unconfirmed suspicions and accumulating evidence. In debugging programs or electronic circuits, a particular object may be described in several distinct ways, and the interaction between these descriptions is significant. In describing scenarios for possible consequences of an action, the number of possibilities is (apparently) much larger than the small number of alternatives to the parallelepiped in Blocks World. All of these problems are poorly understood in any formalism. It will be

valuable to attempt to formulate the issues rigorously within the concept of frames.

References

Particular thanks for their helpful comments and criticisms go to Mike Dunlavy, Carl Hewitt, Gene Freuder, and Marvin Minsky.

Goldstein 1974
Ira Goldstein, "Understanding Fixed Instruction Turtle Programs", M.I.T. Artificial Intelligence TR-294, 1974.

Hewitt 1973
Carl E. Hewitt, et al. "Automating Knowledge Based Programming and Validation Using ACTORS", M.I.T. Project MAC, PLANNER Group working paper, 1973. Chapters of this paper have appeared as papers in 1973 IJCAI and SIGPLAN proceedings.

Lerman & Woodham 1973
Jerome P. Lerman & Robert J. Woodham, The TRACK Program Package, M.I.T. Artificial Intelligence Vision Flash 49, 1973.

Minsky 1974
Marvin Minsky, "A Framework for Representing Knowledge", forthcoming paper, 1974.

Sussman 1973
Gerald Jay Sussman, "A Computational Model of Skill Acquisition", M.I.T. Artificial Intelligence TR-297, 1973.

Winston 1973
Patrick H. Winston, ed. "Progress in Vision and Robotics", M.I.T. Artificial Intelligence TR-281, 1973.