MASSACHUSETTS INSTITUTE OF TECHNOLOGY
ARTIFICIAL INTELLIGENCE LABORATORY

Working Paper 112 October 1975

ASSIGNING HIERARCHICAL DESCRIPTIONS TO VISUAL ASSEMBLIES OF BLOCKS WITH OCCLUSION

Michael R. Dunlavey

Abstract

This memo describes a program for parsing simple two-dimensional piles of blocks into plausible nested subassemblies. Each subassembly must be one of a few types known to the program, such as stack, tower, or arch. Each subassembly has the overall shape of a single block, allowing it to behave as part of another subassembly. Occlusion is represented by an area of the image plane whose contents cannot be seen. Heuristic aspects of the program are concerned with 1) ambiguity among competing subassemblies due to sloppiness of the placement of the blocks, 2) ambiguity due to uncertain measurements of blocks which are partially occluded, and 3) total ambiguity as to the contents of the occluded region.

Choice among competing subassemblies is accomplished by first making a topological description of the network of conflicts among subassemblies, then considering only the simplest competing subset. If this does not clearly indicate a winner, the system can make an in-depth comparison of the internal structures of the last two competing subassemblies.

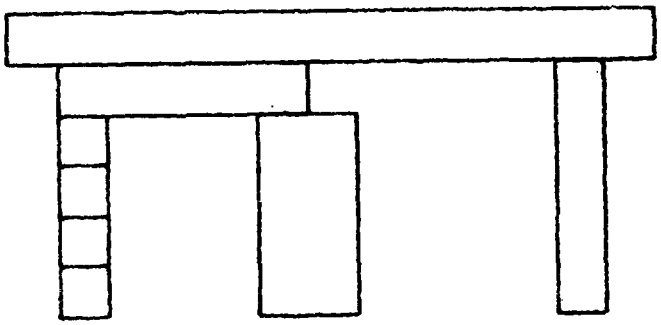
Uncertainty as to measurements of blocks is handled by creation of a disjunction of more certain blocks, each of which participates in the parsing process. If this disjunction results in a pair of competing subassemblies, only one is used, the other being hidden as an alternate to the first, so that the choice of which will be accepted can be deferred. This is a deferrable choice because the alternate subassemblies are so closely similar that the parsing process does not depend on choosing one of them.

Uncertainty due to occlusion is handled by allowing a potential subassembly to use the occluded area as a "wild card", meaning that if the subassembly can be completed by creating a block which intersects the occluded area, it is so completed. Such an imaginary block may later be consolidated with a real one, or it may remain imaginary.

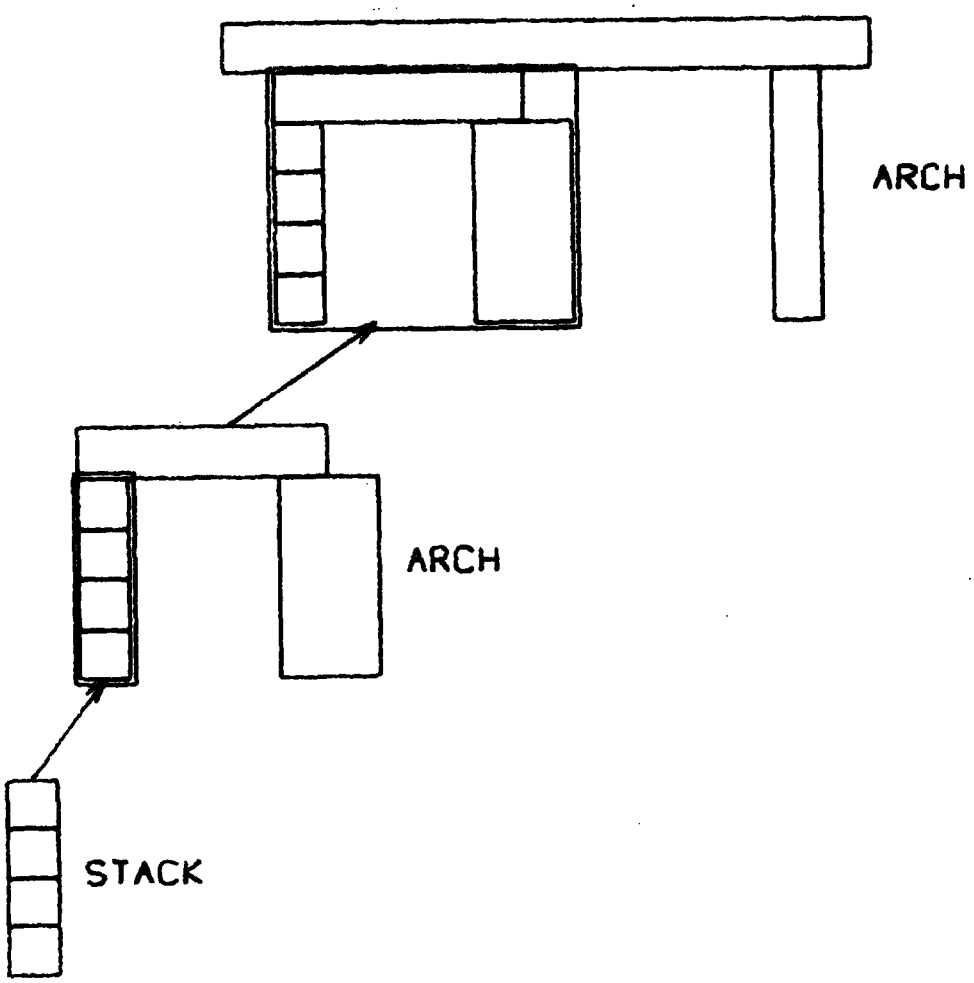
The reason for studying this problem is to become acquainted with the program and data structure needed to assign a nested structural description to a complicated visual assembly in which occlusion makes the data incomplete. The extension to 3-dimensional descriptions should be straightforward.

This report describes research done at the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology. Support for the laboratory's artificial intelligence research is provided in part by the Advanced Research Projects Agency of the Department of Defense under Office of Naval Research contract N00014-75-C-0643.

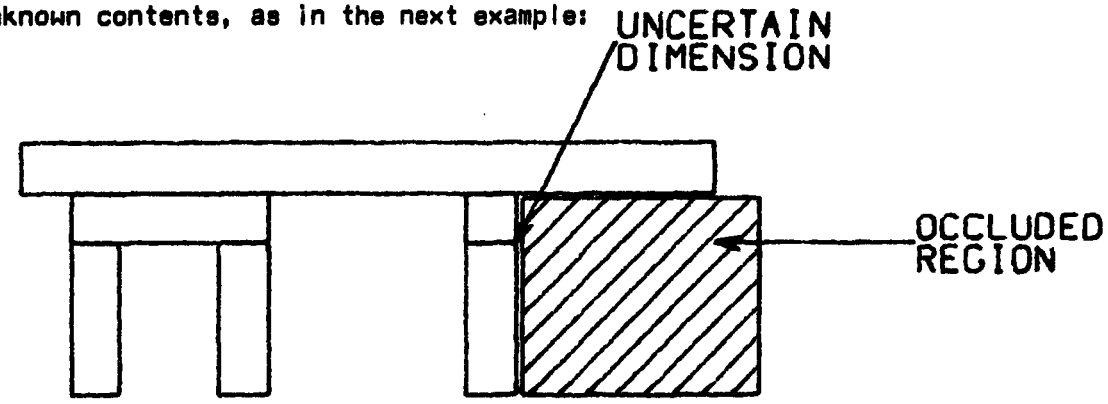
The problem is the following: Given a collection of blocks, represented as rectangles in two dimensions, all aligned vertically or horizontally, the task is to divide the collection into a hierarchy of plausible subassemblies. An example is the following collection of blocks:



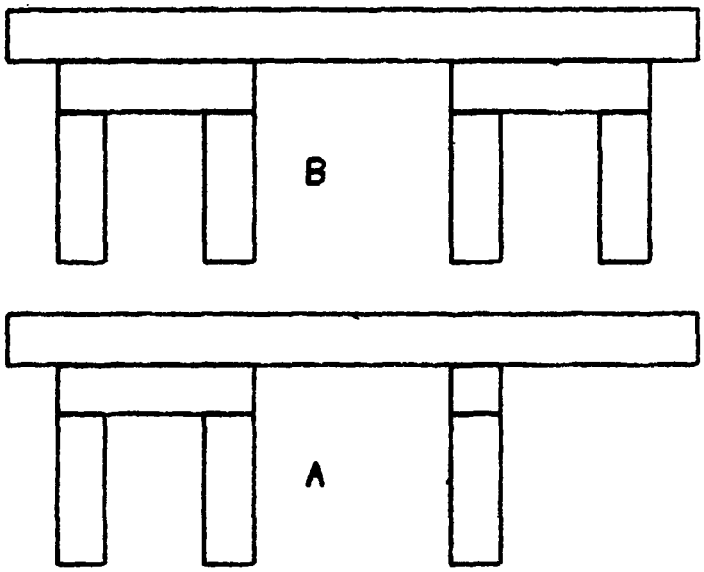
which could be parsed as follows:



Occlusion is simulated by means of an artificial blind spot, a rectangle of unknown contents, as in the next example:



for which at least two parsings are plausible:



with B being more plausible due to symmetry of the scene.

The reason for studying this problem is that it addresses the same issues of program and data structure as in the full three-dimensional problem, while being considerably easier to program. The issues addressed are 1) what is a good parsing strategy when all the input data is known, but when the subassemblies are not always well-formed, and 2) how should ambiguity and expectation interact when some of the input data is known to be unavailable?

Data Structures

The system is a breadth-first bottom-up parser. It is a loop, taking the BLOCKS list as real input, identifying potential objects, choosing a winner, replacing it in the BLOCKS list, and starting over. The principal

data structures for the program are the following:

Parameters -- Parameters take the place of numbers in the descriptions of blocks. A parameter is a list of the form:

(value certainty-flag)

where the value is usually numeric, and the certainty-flag is either

T - known,
NIL - uncertain,
> - uncertain but not lower, or
< - uncertain but not higher.

BLOCKS -- a list of all the blocks currently acting as known input to the parser. Each block may be from the original scene or may be the result of replacing some subassembly. A block is an atom, such as NB-1, having a Lisp value and some properties. The Lisp value is a 4-tuple of parameters:

(min-X max-X min-Y max-Y)

specifying position and size. The block may also have some optional properties:

OB -- if the block stands for an assembly of other blocks, this property contains a list describing that assembly, such as:

((B1 B2) ARCHL 0.3 (NB-1 B1 B2 B3)(B1 B2 B3))

where (B1 B2) is the seed pair from which the arch was found, ARCHL indicates the type of object - an arch discovered from the left side, 0.3 is a crude measure of the well-formedness of the arch, 0.0 being perfect, the sum of internal relationship measures, (NB-1 B1 B2 B3) gives the name of the overall block along with the names of the constituent blocks in a particular order corresponding to the specific parts of the arch, and (B1 B2 B3) is simply a list of all the constituents of the assembly.

ALTERNATE -- if present, points to another block which could be substituted for this one. This is a result of an ambiguity where the same assembly of blocks could be described two different ways. This is a qualitatively less important kind of ambiguity than that between conflicting subassemblies. It is a <u>deferrable choice</u>.

OBS -- a list of potential objects, or subassemblies in the scene. This is generated by running program FINDOBS against BLOCKS. It is a list of hypothetical blocks, each having an OB property describing its constituents and type of assembly. FINDOBS is an optimistic program, so the potential objects that are found usually conflict with one another, by sharing constituents, so the next step is to try to find the best one. This cannot be done on the basis of the well-formedness measure because it is too crude, so first the system attempts to see if there is any recognizable pattern among the conflicts between the potential objects.

CFNETS -- the list of disjoint conflict nets, generated by function GOBBLE applied to OBS. Each conflict net is of the form

(type b1 b2 b3 ...)

where b1 b2 b3 ... is the list of potential objects contained in the conflict net, and where the possible types, in order of increasing complexity are:

NONE - signifying a singleton net (no conflicts),
PAIR - two conflicting potential objects,
CHAIN - signifying a chain of conflicting potential objects, from b1 to the last,
STAR - meaning the conflicts converge upon a central potential object, b2, b1 being one point.
MESS - none of the above.

The least complex conflict net is chosen and stored in variable CFNET. This is then passed off to a specialist for finding the winning potential object out of a conflict net, or possibly generating a new potential object, as in the case of an NSTACK subassembly. The winning potential object then takes the place of its constituents in the BLOCKS list, completing one cycle of the parser, except for possible consolidations.

Algorithms

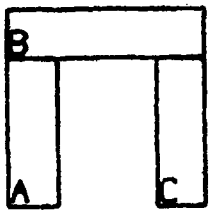
Finding Potential Objects

Program FINDOBS takes a list of blocks considered to "really exist" and returns a list of possible arches and stacks in that set of blocks. It works by calling program SYS on every possible pair of blocks. SYS takes the pair

(a b)

and sees if either they form a stack:

or there is a third block c such that they form an arch:

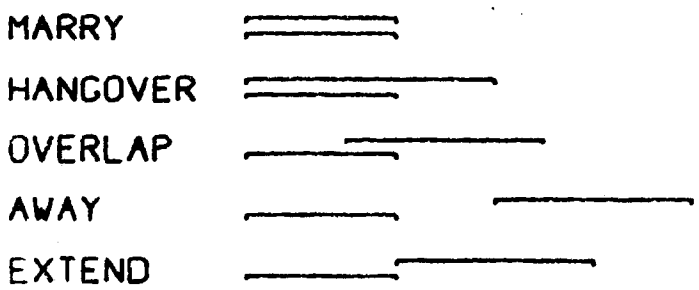


These perceptions are based on numerical-valued spatial relationships,

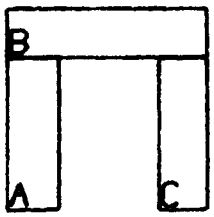
where 0.0 is considered perfect, and 0.25 is typical of a marginal value. A two-dimensional spatial relation between blocks is expressed as a conjunction of one-dimensional relations, those between the projections of the blocks onto the horizontal and vertical axes. For example, for two blocks to be a stack, the vertical projections extend, but the horizontal projections marry:

EXTENDPY(A B), MARRYPX (A B)

The other one-dimensional relationships are:

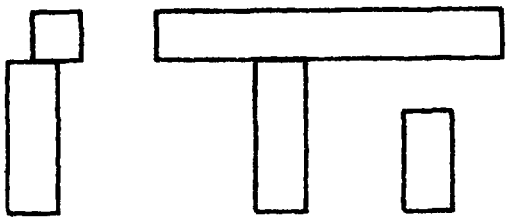


In terms of these relations, an arch has the ideal description:

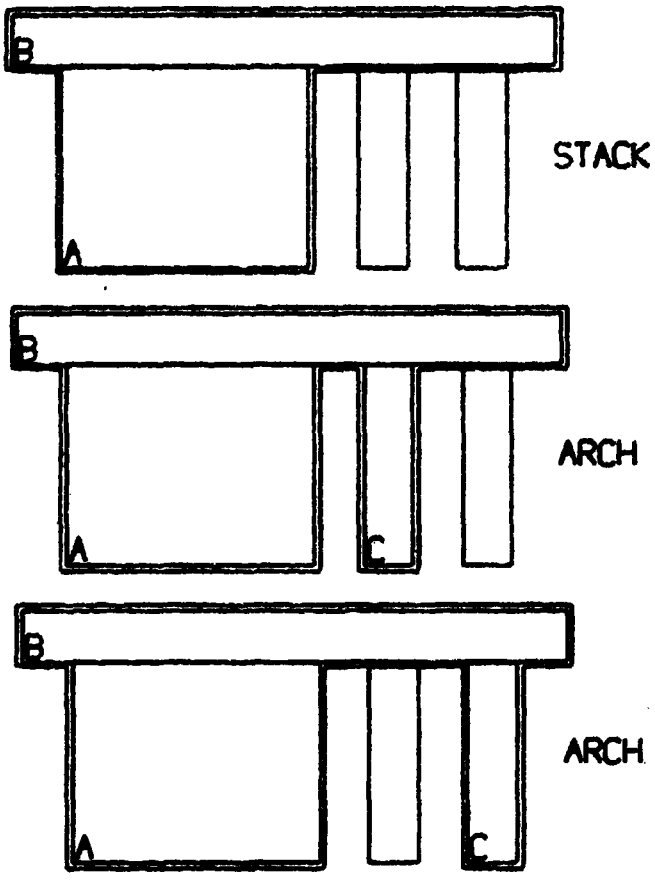


EXTENDPY(A B), HANGOVERPX(A B),
EXTENDPY(C B), HANGOVERPX-(C B),
MARRYPY(A C), AWAYPX(A C)

However, these relationships can be less than perfect, in which case their values will be greater than zero, while still reporting the assembly as a potential object. For example, the following are accepted:



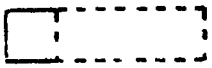
A simple threshold on the sum of the relationships is used to admit the grouping as a potential object. One (a b) pair may result in more than one potential grouping:



SYS and FINDOBS simply report all of these. It is not appropriate to use the numerical value to choose among them because it is too crude. This is handled later by conflict specialists.

If there is no uncertainty in the scene, there is no more to say about what FINDOBS does. If there is uncertainty, it takes two forms:

1) Uncertain dimensions of a block, such as

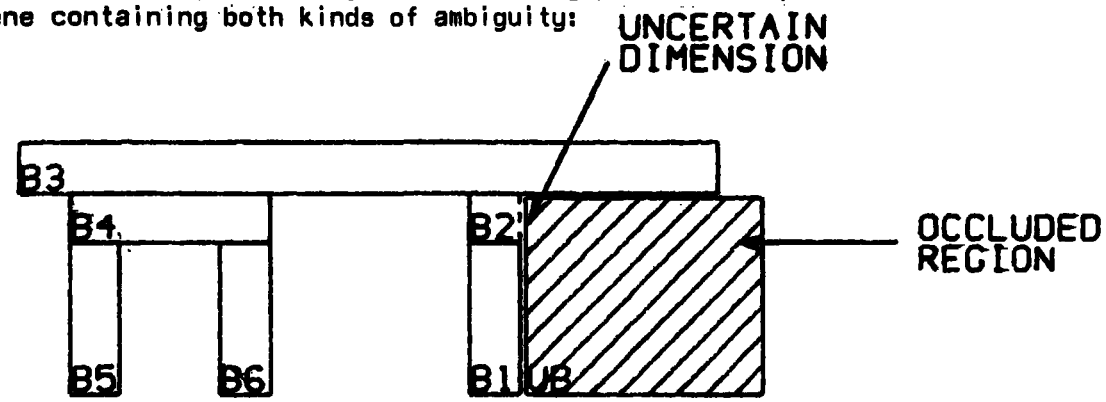


where the max-X value of a block is given, but it might be higher, represented by the parameter:

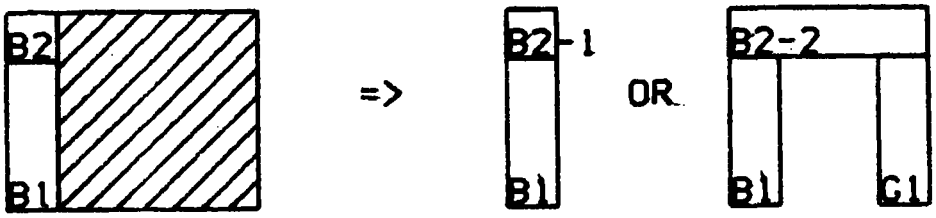
(4.0 >)

2) an occluded area in the scene, represented by a rectangle, which might be hiding anything.

The two kinds of uncertainty are qualitatively different. In the case of an uncertain dimension, for example B2 above, whose max-X value is uncertain, SYS simply generates two blocks, one the same size as B2, and one somewhat longer, and uses these as distinct possibilities instead of B2. These two new blocks, called B2-1 and B2-2 are stored on the SONS property of B2. These are then treated as two separate blocks in place of B2, and SYS behaves for the moment as though these were two separate blocks, possibly resulting in conflicting potential objects. Here is a scene containing both kinds of ambiguity:



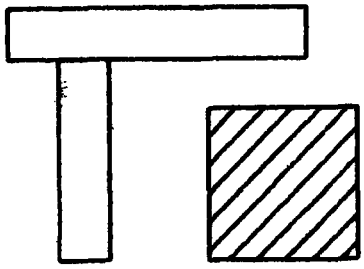
If the ambiguity due to uncertain measurement results in two separate potential objects:



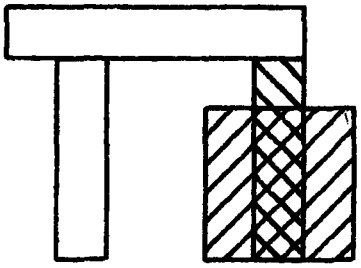
then only one of them is returned, the other being stored under the ALTERNATE property of the first. This is an example of a deferrable choice because each grouping is just about as good as the other for parsing purposes, having the same height and using up the same blocks from the scene, so it is wise to attempt to hide this ambiguity from the subsequent

conflict-resolution phase so as to simplify that task.

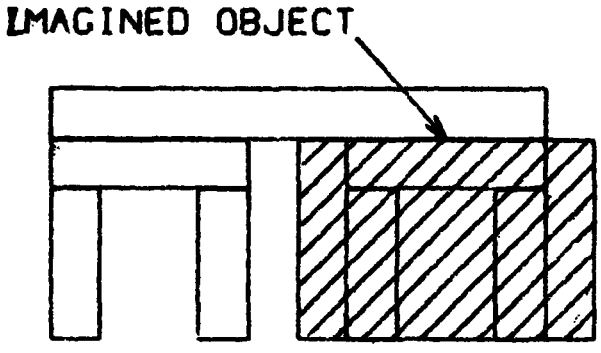
The second kind of uncertainty does not readily predict its alternatives, so SYS waits for a potential object to suggest what is in the occluded area. For example, if SYS is trying to complete an arch, but can find no right leg:



it constructs a hypothetical right leg and sees if it intersects the occluded area:



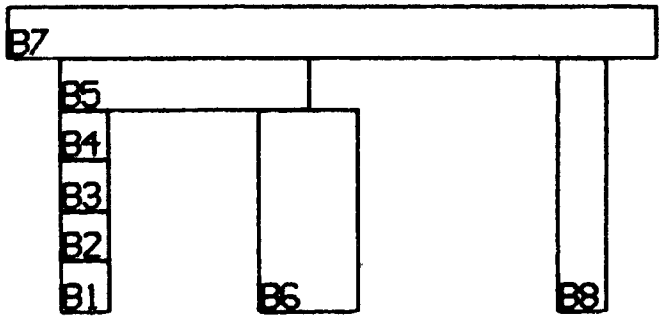
If so, it assumes the right leg exists (with suitably uncertain measurements) and proceeds as though it had seen it. It also assumes the right leg has the same internal structure (if any) as the left leg:



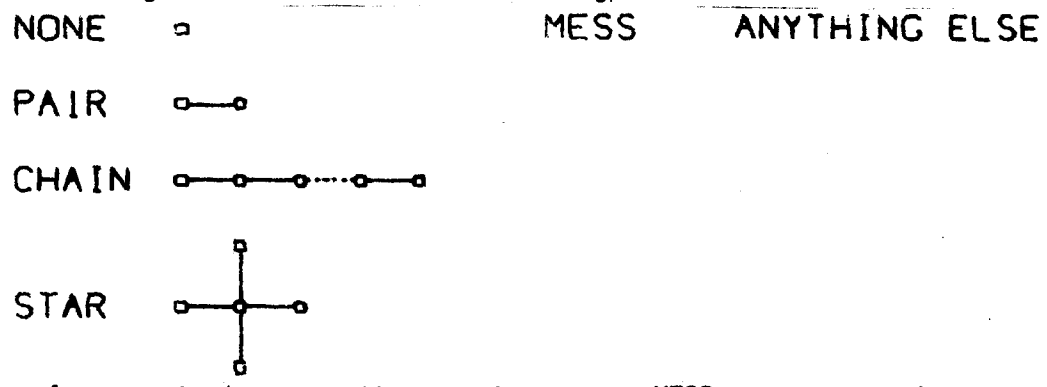
Conflict Resolution

In this program, the strongest form of conflict between potential objects is the sharing of a constituent. This is certainly not an eternal

rule, but is most useful as a first approximation because it has great power to disambiguate. The analysis of conflicting potential objects proceeds in two steps. First the network of conflicts is partitioned into connected subnets, each of which is classified by topological type, and all but the simplest are discarded. For example, in the following scene:



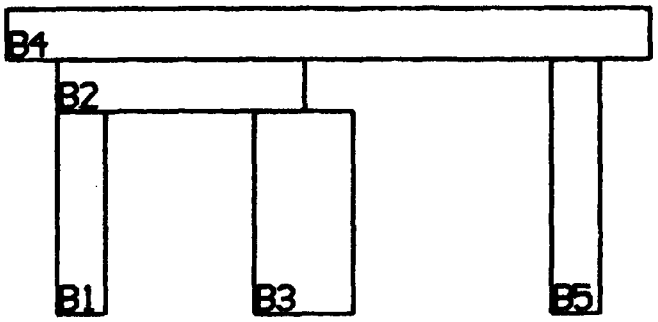
three potential objects are reported by FINDOBS, (B1 B2), (B2 B3), and (B3 B4), forming a chain of conflicts. The types of conflict nets are:



If the simplest conflict net is of type MESS, an attempt is made to reclassify it after first discarding all but the best potential objects on the basis of numerical score. This is the only point at which numerical score is used to discriminate among potential objects. This situation arises if FINDOBS has for some reason been overly optimistic and has found too many potential objects among a small set of blocks. Such a set of potential objects is likely to be almost completely connected by conflicts.

The second phase of conflict analysis is to apply a specialist program to the simplest conflict net found. The specialist returns the winning potential object, or possibly a new potential object. For example, the CHAIN-SPECIALIST normally selects one end or the other of the chain, whichever is more certain (doesn't have an alternate). However, in the case of a chain in which every potential object is a stack, this indicates the presence of a single stack of more than two blocks, or an NSTACK, which is returned as the winning potential object.

In the case of a simple PAIR of conflicting objects, the PAIR-SPECIALIST goes to greater lengths to determine the winner. For example, in the following scene, there are two potential arches:



a1 = (B1 B2 B3) score 0.3

a2 = (B1 B4 B5) score 0.4

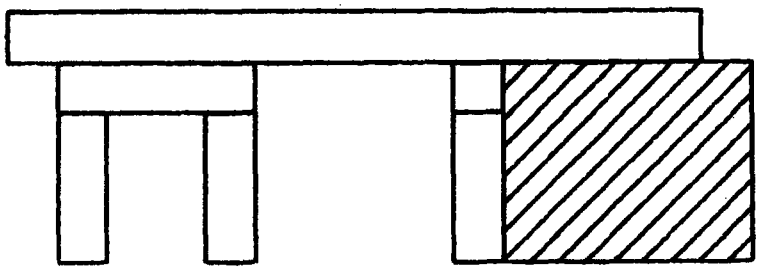
which conflict by having B1 in common. First the conflict block is found, namely B1, along with its name in each arch, that is, a, b, or c. Then, in each arch, the relationships in which B1 participates are computed, and these are compared one by one (where the program already knows the correspondence between relationships).

B1 in a1	B1 in a2
extendpy(a b) = 0.0	extendpy(a b) = 0.2
hangoverpx(a b) = 0.0	hangoverpx(a b) = 0.0
marrypy(a c) = 0.0	marrypy(a c) = 0.2

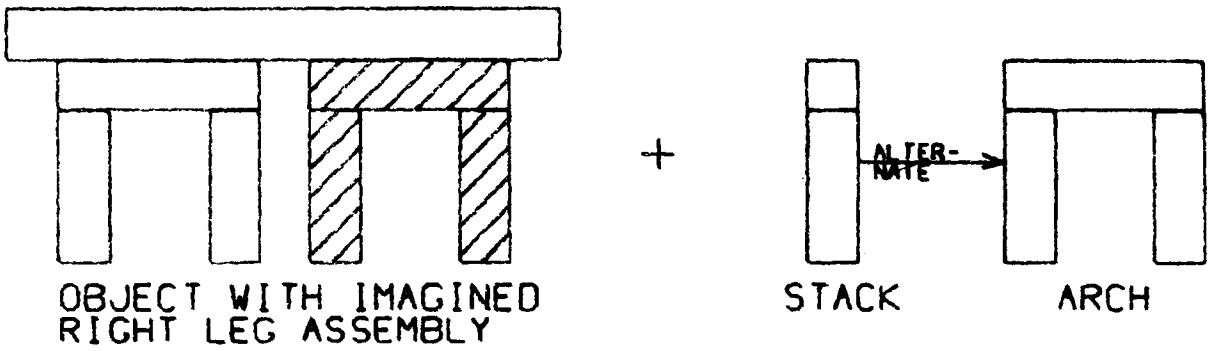
Since there is a clear winner on a relationship-by-relationship comparison, a1 is chosen. This kind of in-depth comparison is necessary rather than using the simple overall scores because those scores are too easily influenced by factors having nothing to do with the conflict, such as the shape of B3.

Consolidating Predicted and Discovered Objects

Sometimes it is necessary to combine objects. In analyzing the following scene:



the final cycle of the system produces the following structure:



The shaded arch was created to complete the overall arch (due to occlusion), before the structure on the right was parsed.

The hypothesized arch is compared with the stack in program UNIT-MATCH. Since they overlap, and since their known measurements are the same, they are considered as matching. Since the stack has an alternate which is of the same type as the hypothetical arch, the alternate is chosen instead. Since the match is successful, a small program is created which will replace all pointers to the imagined arch by pointers to the "real" arch, which program is subsequently executed.