

The ACRONYM Model-Based Vision System

Rodney A. Brooks, Russell Greiner and Thomas O. Binford
Stanford Artificial Intelligence Laboratory,
Stanford University,
Stanford, California, 94305, U.S.A.

ACRONYM is a model-based image understanding system. It demonstrates mechanisms for interpretation of images with generic object classes and generic viewing conditions, in a way that is generalizable. It incorporates a powerful geometric modeling capability with a high level modeling language for natural communication with the user in terms of object models. A user gives high level descriptions of both generic and specific instances of objects. A rule-based inference system produces a viewpoint dependent symbolic summary of the predicted appearance of the objects. This geometric reasoning capability enables the system to incorporate and relate knowledge and information at different levels. This summary drives a powerful syntactic matcher to find instances of the objects in preprocessed images.

1. INTRODUCTION

ACRONYM is a vision system based on powerful geometric modeling capabilities. Its geometric modeling system is of interest in itself. ACRONYM is also the basis for research in programming robots from a data base of parts models.

ACRONYM is intended to deal with key problems of interpreting scenes:

1. Generalizability is a central issue. A photointerpreter performs a broad range of tasks which involve different object classes, which have different contextual information, and which vary greatly at the image level because of varied viewpoint, illumination, sensors, weather, and obscuration and camouflage. Systems for very different tasks should be constructed from a large core of common modules and a small set of task-specific modules. For a single system to map this wide range of task elements onto a common set of modules, it is convenient that the modules represent a natural decomposition of the problem into physically meaningful elements, for example, those we use in our own human description of the problem. Our basis for generalizability is the use of a tightly structured hierarchy of geometric representations.

2. A variety of information and knowledge is available. A photointerpreter solves a puzzle by piecing together selected and multiple clues from current images, background information, and previous images. In doing so, he relies heavily on spatial interpretation from stereo imaging and shadows, and spatial knowledge about structures.

This research was sponsored by ARPA contract MDA-903-76-C-0206 and NSF contract DAR-7815914.

Integrating multiple cues within a single task is a key issue which raises technical questions for representation.

3. It is important that the system be generic with respect to objects and generic with respect to viewing conditions. Our approach to generic interpretation is to use object models made from generic parts, and to use generic predictions of appearances of generic parts.

4. Users should be able to specify tasks in a natural and simple way. Geometric models are natural for both the user and the vision system. Ultimately, users will be able to instruct systems in natural language. The representation hierarchy of ACRONYM is the basis for a Vision Language which could serve as a bridge between natural language and standard programming languages.

We describe capabilities for goal-directed model-based scene interpretation based on mapping from object and part models to predicted appearances to picture structures. ACRONYM is designed to function in a descriptive sense (bottom-up) as well, mapping from picture structures to features of parts to object models. In typical situations, both functions are involved.

Model-based vision has major applications in photointerpretation and manufacturing. Consider two typical tasks for the system:

1. A photointerpreter programs the system to monitor aircraft at an airfield. Aircraft models are entered along with a model of the airfield. The system is told that aircraft are found at airfields on runways, taxiways, or storage areas.

2. An engineer programs ACRONYM to inspect a part, or to pick the part from a bin of identical parts and place it in an assembly. He uses the part model from a CAD system and specifies a model of the workstation.

The diagram in fig. 1 shows the main modules, data structures and data flow paths, along with the local environment of ACRONYM. The inner enclosed boxes represent data structures, the other boxes are program modules. As of January 1979, implementations of all the modules existed, but not all data paths were in operation.

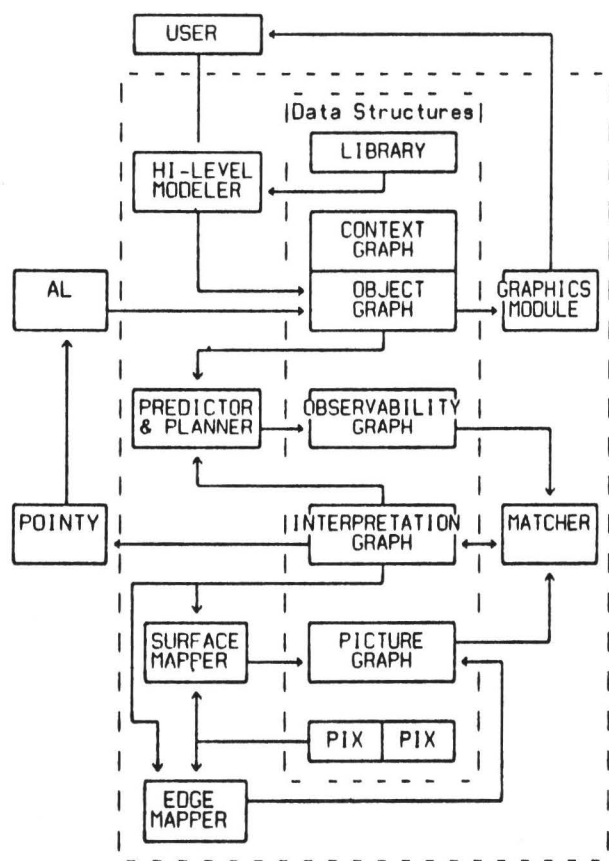


Fig. 1. The ACRONYM system.

The user interacts with a high-level modeler to produce both generic and specific models. There will eventually be a library of useful partial models to draw upon. The result of this modeling process is the Object Graph. This graph drives a graphics module which provides the user with visual feedback of shape and spatial relations during the modeling session. The Predictor and Planner module produces a symbolic summary of the way an object modeled in the Object Graph can be expected to appear in an image. This summary is called the Observability Graph and is used to program the matcher from above. The Matcher

constructs an Interpretation Graph from the Observability Graph and the Picture Graph, which is a symbolic description computed from the image by the surface mapper, and the edge mapper. The edge mapper uses techniques of Nevatia and Babu [23] to extract two dimensional shape descriptions from the images. This is augmented by depth information from the stereo pair of images (when a pair is used) in the surface mapper, using techniques developed by Arnold [2].

Also included in fig 1 are modules for AL and POINTY (see Goldman [13]). AL is a high level language for controlling manipulator arms. ACRONYM has been interfaced to the AL compiler, so that instead of driving the physical arms, AL drives ACRONYM models of arms. ACRONYM can update its display of an arm faster than once per second on a moderately loaded system, giving real time simulation. There is an interface in the other direction to POINTY. POINTY is an interpretive subset of AL, originally designed to be used in a lead and teach mode, to easily obtain spatial information for AL programs. ACRONYM can control the arms via POINTY. So far we have used this interface only for debugging, but it will eventually be used for fully automated assembly tasks, with model-based vision.

2. PREVIOUS MODEL-BASED SYSTEMS

ACRONYM provides a unified approach to many aspects of goal-directed vision, and generalizes the approaches of previous vision systems. It has a richer three dimensional model representation, a rule-based planning system, and makes more extensive use of shape and relational information than previous systems. We briefly survey some existing systems to summarize their use of models and their matching processes.

The MSYS system (Barrow and Tenenbaum [4]) models objects in terms of their height and orientation, and models three dimensional spatial relationships between them in terms of above, below, in front, in back, left and right. Some models (e.g. of a floor) are also constrained to correspond to homogeneous regions within an image. An image is roughly segmented by hand. Simulated range data is used to determine three dimensional locations and orientations of regions and their spatial relations. Horizontal surfaces are identified by height. No shape information is used at all. A matcher matches the data against the model. These simple models are sufficient to identify typical objects in a room scene. This demonstrates that spatial relations can be a strong constraint when identifying regions of an image. However the models used rely on a particular viewpoint, and the ability to propagate constraints throughout the whole image. If some other viewpoint is used, or if the scene is cluttered with objects not contained in the model, the constraint approach will break down. Tenenbaum and Barrow [25] have also used this system to drive the IGS interpretation guided segmentation system which produces segmentations constrained to be consistent with the high level model.

Garvey [12] developed a system which models objects in terms of their height and orientation, three dimensional spatial relationships with each other and the expected ranges of brightness, hue and saturation of their pixels. Approximate indications of size and shape may also be included. In interactive sessions with sample pictures, the system computes histograms of these features over manually segmented regions. Confidence values are given to the spatial relations by the user. The system is given the task of finding a given object in an image. It knows the costs associated with testing pixels for each attribute, and so as an initial coarse pass, it tests pixels randomly selected from throughout the image with tests that will cheaply rule out large areas. These tests are selected by examining the models of known objects for cheap distinguishing features of the desired object. Garvey calls this initial phase acquisition. In the second phase, validation, the models are used to determine more expensive tests, over this reduced area, which will disambiguate the desired object from others which might also have produced the pixels first obtained. Spatial relationships in the models can be used to drive the system when looking for small objects. Larger objects, expected to be adjacent in the image (such as a table top when looking for a telephone) are found first, reducing the search area for the small object. The search for the small object can then be carried out at a much finer level. The planning stage and coarse to fine strategy used improve efficiency but this system too suffers from the inability to use general shape information.

Ballard, Brown and Feldman [3] describe a general purpose system for programming knowledge-based vision tasks. The knowledge or model is in terms of nodes describing expected image features, and spatial constraints between them. Templates are used to describe shapes. The user must code an executive match procedure for the particular task domain. The system has been used for aerial image interpretation and finding ribs in radiographs of human chest cavities. While very general purpose, the system is heavily dependent on using a fixed and known viewpoint, and requires detailed programming for each new domain. Furthermore the user must model objects in the image domain rather than the three dimensional domain.

Bolles [9] has a Verification Vision system which relies only on the three dimensional relationships of observables to locate a mechanical part accurately within an automatic assembly work station. The position of the part is known in advance to within a few inches, and its orientation to within about fifteen degrees. Thus the models can be very viewpoint dependent. Rather than use shape descriptors, Bolles relies on very local operators such as edge detectors, blob characterizers, and Moravec's [20] interest operator to characterize the observables. Due to the highly constrained nature of the task these operators have a reasonable chance of matching the correct feature in the image. The system uses a generalized least squares algorithm, and clique finding to match the model of the known spatial relationships of the object to the features identified by the operators. The model can be used to drive a further

refinement phase, where the location and uncertainty region for further features are predicted. These features can then be searched for in the restricted region, with much reduced probability of incorrect identification. Notice that the models used here are simply a collection of points on the surface of the object.

Kanade [16] distinguishes the image domain, the domain of observable facts from viewing the scene in either intensity or range data, and the scene domain, where objects are modeled. He uses a 2 1/2 D scene domain. Objects are represented as image regions. Shape and spatial relations describe the regions. Objects have multiple representations for multiple viewpoints, but these must be explicitly described by the user. The matching process tries to match observed patches against modeled patches.

Rubin's [24] ARGOS system stores multiple representations of buildings, in terms of such things as texture, color, orientation and gross shape features, all gleaned from training examples. It also has three dimensional knowledge of the positions of the buildings, which is translated into adjacency information to guide the search for labellings of pixels. The search technique is a very local pixel (or segment) based "Locus" search. This localness means that adjacency is the only meaningful relation which can be used.

These systems make valuable contributions to model-based vision, however they do not generalize in significant ways, with respect to viewpoint, with respect to object classes, and with respect to very large classes of objects. In almost all cases, data and model are matched at the same level, e.g. surface to surface or point to point. ACRONYM adds the capability to reason between different levels of representation, based on a hierarchy of representations.

3. MODELING

The Object Graph contains three dimensional models. It provides volumetric representations of objects, both specific and generic, and three dimensional spatial relations between volume elements. It also provides for multiple levels of representation of objects, from coarse to fine. The basic volume primitive used is the generalized cone, first introduced by Binford [5]. Spatial relationships of volume elements within an object are defined hierarchically. Both specific and generic volume elements, and relations between them can be modeled. Thus an airplane can be defined to have exactly two wings, while an airport can be described as having from one to four runways.

The Context Graph is not really distinct from the Object Graph. It is only a convenience to talk about objects such as aircraft and their context, airfields. Both use the same representational mechanisms.

3.1 Generalized Cones

Our models are part/whole graphs; the models resemble

stick figures. Parts have subparts, they are subgraphs whose leaves are our primitives, generalized cones. Levels of detail correspond to levels of the part/whole subgraph. A natural and useful part/whole segmentation depends on the primitives used for representation of parts. Design criteria for representation were presented in Thomas and Binford [26].

A generalized cone is defined by generalized translational invariance. It is described by three sub-primitives; a spine, a cross section and a sweeping rule. In its most general form, a generalized cone is the volume swept out by the cross section as it is translated along the spine, while being deformed according to the sweeping rule. Elongation is not a necessary property of generalized cones; neither is a circular cross section.

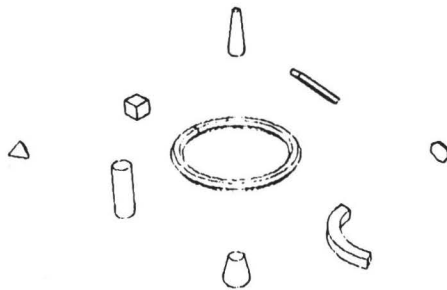


Fig. 2. Some Generalized Cones

Most objects can be segmented in a few ways into volume elements whose cross sections change smoothly along some space curve (see [21] or Nevatia and Binford [22]). These volume elements can then be modeled by a single generalized cone, with its spine along that space curve. Generalized cones accurately represent a variety of fabrication processes. Intrinsic properties of a generalized cone can be approximately inferred from its two dimensional image under a broad range of viewing conditions (Nevatia [21], Marr and Nishihara [18]). For example, the length to width ratio varies approximately as the sine of the angle between the cone axis and the ray from the axis to the observer. While generalized cones need not be elongated, interpretation is much simpler for elongated objects and current techniques rely on elongation. Marr [17] has shown that interpretation of the occluding contour of an image of an object, under certain restrictive assumptions, is equivalent to interpretation as generalized cones. Thus generalized cones correspond to a natural decomposition of large classes of objects, and they are easy to match to a two dimensional image. Moreover they provide a compact representation for complex volumes. The representation of complex objects requires representation of only a few parts, each of which is about as complex as the representation of a cube.

Generalized cones have been used in a number of vision and geometric modeling systems. Agin [1] built generalized

cone descriptions from images of simple objects such as a snake and a hammer. Nevatia [21] recognized images of dolls and toy horses, by indexing into a library of generalized cone models. Miyamoto and Binford [19] built a general modeling and graphics display system based on generalized cones. Hollerbach [14] modeled and recognized classes of Greek vases. Marr and Nishihara [18] described a method to determine the orientation of objects by matching them to generalized cone models.

Representation of spherical volumes as spheres is intuitively better than representation as generalized cones, especially for portions of spheres. We set out to incorporate two primary geometrical constructions; translation and rotation. Generalized cones are based on generalized translational invariance. Spheres are based on rotational invariance. We have a straightforward generalization of rotational invariance, however generalized spheres, as we understand them now, do not seem as powerful an extension as generalized cones, and we require more work to include them in our representation. We have also thought it necessary to represent surfaces, obviously to represent cross sections of cones, but also for nonplanar sculptured surfaces such as wings. Generalized translational invariance is useful here, too. Generalized cones specialize to ribbons, which are defined by an axis and a cross section, both of which are space curves not plane curves, and a sweeping rule (see Brooks, Greiner, and Binford [7]). It is not our intention to ignore standard surface spline representations, but rather to explore the advantages and cost of an alternative representation.

We use a more general class of generalized cones than previous systems. However there are still major restrictions. The cross sections must have a boundary which can be decomposed into straight line segments and circular arcs. The cross section can be kept at any constant angle while being swept along the spine. The sweeping rule must be piecewise linear, and continuous. The spine must be continuous and made up of straight line segments. Circular arcs can also be used as segments of the spine, so long as the sweeping rule is constant, the cross section has a piecewise linear boundary and is kept normal to the spine. Fig. 2 shows some examples. This latter class allows for convenient representations of curved roads or taxiways. Analytic work is being done to formalize a natural and adequate extension of this subclass of generalized cones.

The parameters of a cone can describe ranges of possible values, or provide a predicate to test the validity of a proposed value. For instance a generic runway can be described in terms of allowable ranges of length and width - possibly dependent on each other.

3.2 Model Structure

As in previous modeling systems based on generalized cones (e.g. Miyamoto and Binford [19]) there is a local coordinate system for each volume primitive in a model, rather than a viewer centered, or object centered coordinate system. A complex object which is modeled by more than one

generalized cone has an affixment tree (in general a directed graph - but multiple affixments can be inconsistent), where the coordinate system of a cone is that of its parent, modified by the coordinate transform attached to the arc joining them. As in previous systems (see Thomas and Binford [26]) our models also have a hierarchy of level of detail. However we have chosen to separate this hierarchy from the attachment hierarchy.

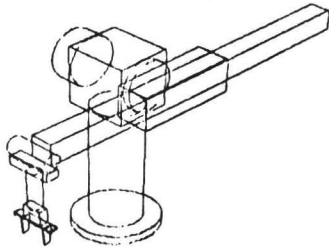


Fig. 3. The Stanford Arm

The following two examples indicate how and why this separation can be useful. It is natural to affix the two wings of an airplane to the fuselage, to describe the spatial relation among them. However wings appear almost the same size as the fuselage, and the two of them combined (especially when an airplane on the ground is viewed from above) provide an axis of elongation of the same order as that given by the fuselage. Thus at the first level of decomposition, we model an airplane as the generalized cones of the fuselage and the two wings, rather than as a single cone. Consider now the Stanford robot arm shown in fig. 3 (this is a drawing produced by ACRONYM from its model of the arm). A natural first level of decomposition of the arm is the upright base, the boom and the hand assembly. The point of interest here is the relation between the boom and the hand. The spatial relation between the boom and the wrist is best described by a series of spatial links through subparts of the hand.

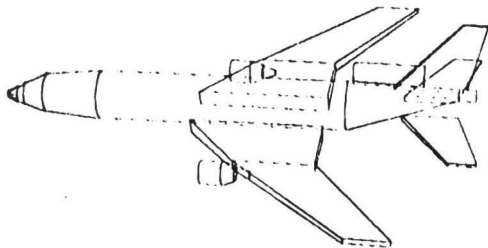


Fig. 4. An L1011

It is important to represent both generic objects (loosely classes of objects) as well as specific objects. For instance we may want to look for airports in general, not only to count the airplanes at each, but also to count the numbers of particular types of airplanes which can be identified. Thus

a DC-10 say, would be recognized as an instance of the Object Graph node airplane, or as a DC-10. An aircraft with special radar would be identified as a modified form of a prototype. If the Object Graph doesn't include a detailed model of a Boeing 767, it might be identified as an airplane, but fail to match precisely any detailed model. A specific model may be a more refined version of a generic model and a specific model can be an example included in a generic model. The modeling system provides language constructs for using a node of the object graph as a prototype for the construction of a new node. This can be done at all levels - from an object down to a sweeping rule.

3.3 Geometric Editor

The high level modeler provides a high level language which gives the user the ability to manipulate models with symbolic names, and provides ways to examine partially built models. There is also a graphics module to help the user with detailed modeling of specific objects. It provides the user with visual feedback of what is being modeled and where the local coordinate systems are. The class of cones allowed can generate complex curved surfaces, e.g. the surface generated by a circle swept along a spine at some non-normal angle while varying in size. Rather than approximate these surfaces with planar surfaces, we manipulate its symbolic description right up until the time to deposit the visible lines in a raster buffer.

We have fast analytic solutions for the appearances of the occluding contours of these surfaces, and have implemented them in a hidden back surface algorithm. It works directly from the object graph representations. Early details of that algorithm appeared in our paper [7]. Work is proceeding on extending this to a full hidden surface algorithm and the full solutions will appear in a later memo. Figs 2, 3, 4 were produced by ACRONYM, requiring approximately 250 milliseconds for the most complex of those figures.

4. PRODUCING THE OBSERVABILITY GRAPH

Given models of objects, ACRONYM attempts to find instances of the objects in images. The Observability Graph will tell the matcher how to find instances. It is a symbolic summary of the expected appearance of objects in the image. It contains generic and specific predictions about shape elements and relations between them, with information about how to find them, what conclusions to draw if identified and what to conclude if they are not there. If information about viewing angles, distance and conditions is available, it can be used to produce more definite predictions. The Predictor and Planner module is a rule based system which uses the Object Graph to produce the Observability Graph.

The program must choose features of the object which correspond to image and surface properties which segmentation programs can find. We will call such features "observables". The features which we are using first include shape and two dimensional spatial relations of shapes

within the image. When dealing with a stereo pair of pictures we also use three dimensional spatial relations and surface information. We intend to extend the class of features after our initial experiments.

4.1 Observables

Since the exact disposition of objects in the image is not known in advance, the Observability Graph can not contain an exact prediction of what will be seen. Rather, it must consist of predictions which adequately describe a range of possible appearances, generic with respect to both object class and viewpoint. They can best be thought of as supplying constraints on the way the picture can be expected to look. For instance, given that the system will be examining aerial photographs of airplanes on the ground, the Predictor and Planner can tell the matcher that when a candidate fuselage is found, wings should be found adjacent to it, with bilateral symmetry. There will be two possibilities for the relative angle, and once the actual angle has been found, the front and rear of the fuselage will have been distinguished. Then the finer task of locating the rear stabilizers (for positive identification of airplane type) can be carried out, confined to a small part of the image. How these are implemented is described in the next paragraph.

The best sort of observables to put into the Observability Graph are those which are invariant. This is a strong requirement; other predictions will be true under a wide range of viewpoints. We will call these quasi-invariant observables. Some are quasi-invariant with respect to object class, while others are quasi-invariant with respect to viewing conditions. For example all airplanes have a long cylindrical generalized cone as the fuselage (invariant with respect to object class), and from most viewpoints (especially aerial views of airplanes on the ground) the fuselage will appear as an elongated ribbon (invariant with respect to viewing conditions). Observables which are functions can be viewed as quasi-invariants too. For example, the orientations of the wings within an image are functions of the orientation of the fuselage.

The Observability Graph itself consists of nodes, arcs and relations. The nodes are either direct shape descriptors or recursively complete Observability graphs themselves. For instance an airport Observability Graph will contain nodes for runway and taxiway shape descriptors, and a node which is itself a complete Observability Graph for the generic class of airplanes. Each node corresponds to something which should be identifiable in an image. Arcs are binary relations which should hold between instantiations of the nodes they link in the image. For instance an arc might say that any instance of a runway should intersect at least one instance of a taxiway. The Observability Graph also allows n-ary relations - called simply relations. These are selected from a library of relations each of which is a function on a list of Observability Graph nodes and their candidate instantiations. Proximity and connected are currently implemented.

4.2 Control of Rules

The Predictor and Planner is clearly a critical module of the ACRONYM system. ACRONYM is the first vision system to incorporate a general reasoning system. It is necessary because we wish to predict the appearance of generic objects, from generic viewpoints. This is a difficult task which requires reasoning about a large body of diverse knowledge. We have chosen to use a rule based system to facilitate experimentation, and to provide additivity of new knowledge. We will first describe the control structure we have implemented for our rules, then discuss some of the techniques used in our early sets of rules to produce the Observability Graph.

The basic control strategy we have chosen is to have our rules consequent driven (i.e. goal directed, or backward chained). The rules can be interpreted to mean that the consequents should be asserted if the conjunction of the antecedents can be proved. This simple strategy is not quite sufficient for the task at hand, however. It is possible that once invoked and "fired" a rule may take control for a while, doing some forward reasoning, and possibly setting up new goals which are attempted before control is handed back to the original invoking mechanism. Davis, Buchanan and Shortliffe [11] use consequent driven rules in the MYCIN medical diagnosis system. Our first implementation of the rule based Predictor and Planner closely followed the MYCIN model. However, we are working with more complex, structured data, and have gradually moved to a rather different system.

Our rules have three components; premises (these are the antecedents), side effects and consequents (possibly multiple consequents for a single rule). Rules are indexed on their consequents. The backward chaining strategy finds all rules which might possibly satisfy the current goal. They are tried in turn until one succeeds. A rule is tried by recursively trying to satisfy its premises, until either one fails (whence the rule is discarded and the next rule tried), or until all premises have been satisfied. In this case the side effects are carried out, and then the consequents are asserted - satisfying the original goal. As the name suggests, the side effects of a rule have nothing to do with the basic goal directed control structure.

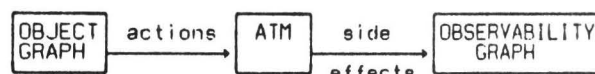


Figure 5. Producing the Observability Graph.

Fig. 5 shows the flow of information from the Object Graph to the Observability Graph. The assertion of the consequents places them into Associative Triple Memory (ATM). The general form of a consequent is a triple consisting of an object (not in the Object Graph sense), an attribute and a value. The attribute is an S-expression - currently these are used purely syntactically via tests for equality, but later semantic information may be explicitly

attached; the object maybe a structured fragment of the Object Graph or some entity constructed by the rules. The actual Observability Graph is produced purely by the side effects of rules firing. The goals which direct the control structure are to place given triples in the ATM. For instance the top level goal for producing the Observability Graph for a model called AIRPORT would be to achieve the triple <AIRPORT, (OBSERV GRAPH), T> in ATM.

In producing the observability graph the Predictor and Planner is guided by two classes of quasi-invariants. Object class invariants broadly determine the node structure of the Observability Graph. The viewing condition invariants determine the detailed contents of the nodes, and the arcs and relations between nodes. Of course this distinction is sometimes blurred. For instance a mixture of the two types of quasi-invariants directs production of the arc which says that runways should intersect a taxiway in the image. The object model says that they should be connected, and a rule which knows that connectivity is invariant under all viewing angles deduces that intersection of the regions corresponding to the nodes will be observable.

5. MATCHING

The Matcher uses the Observability Graph generated by the Predictor and Planner to locate an object in a Picture Graph. The Matcher makes an Interpretation Mapping which maps the Observability Graph into the Picture Graph. It uses a goal-directed matching scheme, searching for features indicated in the Observability Graph. It uses relaxation techniques to resolve constraints iteratively. To improve efficiency, the Matcher works in a coarse to fine order which is directed by the Observability Graph. For example, in identifying aircraft, the Observability Graph has several levels of detail, first the aircraft context, an airfield, then the top level of detail of the aircraft, fuselage and wings, then the finer level of detail which distinguishes individual aircraft, i.e. engine number and placement. The Predictor and Planner uses the level of detail of the Object Graph and observability criteria to constrain the Matcher's search space.

Nodes of the Picture Graph correspond to ribbons, surfaces, and curves. Its arcs and relations indicate spatial relations which hold between sets of nodes. We currently use intersection, colinearity, and containment arcs. The classes of nodes and arcs are open-ended to accomodate other primitives and relations. Ribbons and curves are obtained from an Edge Mapper (see Brooks [6]) which uses a line finder of Nevatia and Babu [23]. Surfaces will be obtained from a Stereo Mapper by Arnold [2].

The Matcher attempts to instantiate each node of the Observability Graph by a set of Picture nodes which satisfy properties of the Observability Node. These local interpretations map a part of the Observability Graph into a limited part of the picture. Context is expressed as Observability arcs and relations among sets of Observability nodes. Local interpretations consistent with

Observability arcs form the initial stage of the Interpretation Graph. For example, a segment of a picture may appear to be a runway when it is considered in isolation. However, this interpretation is not likely if the proposed runway connects with a freeway system, or has a flow of automobiles upon it. This information must be encoded in the Observability Graph by the Predictor and Planner.

Each test evaluated by the Matcher has a pair of procedures, (provided by the Predictor and Planner) one for success and one for failure which are invoked appropriately. One possible action of a feature-test is to cause this match to fail unconditionally - thereby removing the suggested instantiation (e.g. a particular Observability node to Picture node mapping) from consideration. Usual actions simply modify the bookkeeping associated with the "goodness" of the match. Standard conditions such as Must-Be or Should-Not-Be can be encoded as simple special cases.

The Observability Graph can also specify ways to explain a failed condition. More generally, it can specify consequences of interpretations. For example, an interpretation of a ribbon as a fuselage determines the location of the wings. A conjecture may be made which seems plausible at a preliminary stage of the match. If the conjecture is later proven to be false, the conclusions it fostered are re-evaluated, and possibly rejected as well. This follows the "Wait and See" philosophy used throughout the matching process.

In the Picture Graph, a set of edges may be structured as ribbons in several incompatible ways. An Observability node may map to multiple Picture nodes and a Picture node may map to multiple Observability nodes. The mechanism for enforcing compatibility is uniform at all levels.

The Matcher has storage and retrieval facilities for finding predesignated common subexpressions. It is probably too costly to automate the finding of common subexpressions in the Matcher. The process could be partially automated at the level of the Observability Graph or Object Graph, but is probably ineffective if done in a syntactic way. The problem is NP-complete.

The same basic format and evaluating scheme is used by the Observability nodes, arcs and relations as by the full Observability Graph itself. An node may further place certain requirements on the arcs and relations in which it participates. This may be used to specify the number of instances of a given arc in which some node may participate - for example, that a runway should be parallel to at most one other runway in a given airport. This quantification is simple to state and to check. Far more complex conditions could be enforced with this facility. Another feature is the use of a plethora of demons for a variety of cases. There is a growing library of routines, including a transitive closure and a clique finder, which can be used by the Predictor and Planner as appropriate. The range of control structures representable in the Observability Graph, and usable by the Matcher currently far outstrips the ability of the

Predictor and Planner to produce them.

An important aspect of the Matcher is its total ignorance. It is a driver function, designed to "evaluate" the Observability Graph in the context of the Picture Graph. All the specific functions, ordering schemes, demons, etc. are contained within the Observability Graph itself. The only routines in the Matcher, proper, are those which access the relevant data, and which know something about the graph structure used. Extensive revisions are planned for the matcher.

6. CONCLUSIONS

The implementation phase has been underway for approximately one year. At the time of writing, ACRONYM had been tested on an example generated by hand from a real image and is being tested on an example generated completely automatically from real images. The High-level Modeler, the Graphics Generator and the Matcher have been tested extensively and are complete for the full-fledged test without further modification. The Predictor and Planner control scheme is complete - it remains to write more rules encoding the details of transforming the Object Graph into an Observability Graph.

The Matcher has been tested on a hand coded Observability Graph, matching against a hand coded Picture Graph. This test was used during the debugging phase, and was designed to test all modes of operation of the Matcher. Thus the particular Observability Graph used is more complex than can be reasonably be expected to be generated by the Predictor and Planner, at least until we have had considerably more experience. The Matcher was programmed to perform scene labeling for a trihedral blocks world. The nodes of the handcoded observability graph described the possible junctions in a two dimensional line drawing, following the schemes of Huffman [15] and Clowes [10]. The arcs constrained the interpretations of the two junctions defining a line to give it the same labeling.

The High-level Modeler has been used to construct a large number of models. Figs 3 and 4 show output from the Graphics Generator of two objects modeled in the high level language accepted by the Modeler. Many additional models of both industrial parts and airport related scenes have been built with the system. The Modeler and Graphics controller are completely interactive, enabling the user to "fly" around airports or manipulate the robot arm.

The Predictor and Planner has been run on both the arm model displayed in fig. 3, and a generic model of an airport. It produced the complete node structure of the Observability Graphs in both cases, and included quantifiers in the airport case, to describe the numbers of allowable numbers of runways and taxiways in a valid airport instance.

In April 1979, the ACRONYM system was tested with a

simplified model of an L1011, and an aerial photograph of an L1011 at San Francisco airport. It identified an aircraft by locating the fuselage, and two wings without help. This was a very preliminary coupling of many of the modules of ACRONYM (all were involved except the surface mapper which is not yet implemented). Much more development needs to be done to make the system robust.

Work is proceeding on a smarter geometric editor. It will be a knowledge-based editor, which knows about the sorts of operations a user typically wants to perform. It will thus be able to act as an intelligent assistant to a user. So far, neither the problems of the intersection of generalized cones, nor a full hidden surface algorithm have been solved because development of the vision system had higher priority than development of the modeling system. These problems will be tackled in conjunction with the smart editor.

The matcher that is implemented is very powerful, allowing complex quantifications in its input language. The Predictor and Planner does not yet make full use of its capabilities. We intend to look for a formal description of the matcher, perhaps modifying it to fit a clean definition of capabilities. This will give a better handle on the type of rules we should write for the Predictor and Planner and give an estimate on the ultimate capabilities of the ACRONYM system as a whole.

REFERENCES

- [1] Agin, Gerald J., "Representation and Description of Curved Objects," Stanford AIM-173, Oct. 1972.
- [2] Arnold, R. David, "Local Context in Matching Edges for Stereo Vision," *Proc. ARPA Image Understanding Workshop*, Cambridge, May 1978, 65-72.
- [3] Ballard, D. H., C. M. Brown and J. A. Feldman, "An Approach to Knowledge-Directed Image Analysis," *Proc. IJCAI-77*, Cambridge, Aug. 1977, 664-670.
- [4] Barrow, Harry G. and Jay M. Tenenbaum, "MSYS: A System For Reasoning About Scenes", SRI AI Center, Tech. Note 121., March 1976.
- [5] Binford, Thomas O., "Visual Perception by Computer," Invited paper at *IEEE Systems Science and Cybernetics Conference*, Miami, Dec. 1971.
- [6] Brooks, Rodney A., "Goal-Directed Edge Linking and Ribbon Finding," *Proc. ARPA Image Understanding Workshop*, Palo Alto, Apr. 1979, 72-78.
- [7] Brooks, Rodney A., Russell Greiner and Thomas O. Binford, "A Model-Based Vision System," *Proc. ARPA Image Understanding Workshop*, Cambridge, May 1978, 36-44.

- [8] Brooks, Rodney A., Russell Greiner and Thomas O. Binford, "Progress Report on a Model-Based Vision System," *Proc. ARPA Image Understanding Workshop*, Pittsburgh, Nov. 1978, 145-151.
- [9] Bolles, Robert C., "Verification Vision For Programmable Assembly," *Proc. of IJCAI-77*, Cambridge, Aug. 1977, 569-575.
- [10] Clowes, Maxwell, "On Seeing Things," *Artificial Intelligence* 2(1971), 79-116.
- [11] Davis, Randall, Bruce Buchanan and Edward Shortliffe, "Production Rules as a Representation for a Knowledge-Based Consultation Program," Stanford AIM-266, Oct. 1975.
- [12] Garvey, Thomas D., "Perceptual Strategies For Purposive Vision," SRI AI Center, Tech. Note 117, Sep. 1976.
- [13] Goldman, Ron, "Recent Work with the AL System," *Proc. of IJCAI-77*, Cambridge, Aug. 1977, 733-735.
- [14] Hollerbach, John, "Hierarchical Shape Description of Objects by Selection and Modification of Prototypes," MIT AI-TR-346, Nov. 1975.
- [15] Huffman, David, "Impossible Objects as Nonsense Sentences," in *Machine Intelligence* 6, B. Meltzer and D. Michie eds., Edinburgh University Press, Edinburgh, Scotland, 1971.
- [16] Kanade, Takeo, "Model Representations and Control Structures in Image Understanding," *Proc. of IJCAI-77*, Cambridge, Aug. 1977, 1074-1082.
- [17] Marr, D., "Analysis of Occluding Contour," MIT AIM-372, Oct. 1976.
- [18] Marr, D. and H. K. Nishihara, "Representation and Recognition of the Spatial Organization of Three Dimensional Shapes," MIT AIM-377, Aug. 1976.
- [19] Miyamoto, Eiichi and Thomas O. Binford, "Display Generated by a Generalized Cone Representation," *IEEE Conference on Computer Graphics and Image Processing*, May. 1975.
- [20] Moravec, Hans P., "Towards Automatic Visual Obstacle Avoidance," *Proc. of IJCAI-77*, Cambridge, Aug. 1977, 584.
- [21] Nevatia, Ramakant, "Structured Descriptions of Complex Curved Objects for Recognition and Visual Memory," Stanford AIM-250, Oct. 1974.
- [22] Nevatia, Ramakant and Thomas O. Binford, "Description and Recognition of Curved Objects," *Artificial Intelligence* 8(1977), 77-98.
- [23] Nevatia, Ramakant and K. Ramesh Babu, "Linear Feature Extraction," *Proc. ARPA Image Understanding Workshop*, Pittsburgh, Nov. 1978, 73-78.
- [24] Rubin, Steven M., "The ARGOS Image Understanding System," *Proc. ARPA Image Understanding Workshop*, Pittsburgh, Nov. 1978, 159-162.
- [25] Tenenbaum, Jay M. and Harry G. Barrow, "Experiments in Interpretation-Guided Segmentation," SRI AI Center, Tech. Note 123, Mar. 1976.
- [26] Thomas, A. J. and T. O. Binford, "Information Processing Analysis of Visual Perception: A Review," Stanford AIM-227, Jun. 1974.