

AN APPROACH TO KNOWLEDGE-DIRECTED IMAGE ANALYSIS

D.H. Ballard, C.M. Brown, J.A. Feldman
Department of Computer Science
The University of Rochester
Rochester, NY 14627 U.S.A.

Abstract

A vision system is described which uses a semantic network model and a distributed control structure to accomplish the image analysis process. In our view, the process of analyzing an image involves directing attention through geometrical constraints, and the application of procedures attached to different conceptual nodes in the network. Several procedures are typically provided for each conceptual node because, e.g., there may be a need to find the same object in the image in very different contexts. These procedures each have a description containing performance measures and pre- and post-conditions. The descriptions are examined by an executive procedure to determine the procedure that is most useful in a context. The result of analyzing a particular image is a sketchmap, which is an instantiation of the relevant part of the larger generic network model.

The current system attempts to resolve several of the fundamental problems in vision. The semantic network facilitates the interplay between geometric and other relational constraints which are used to direct and limit search. The use of multiple attached procedures allows the modelling of objects under radically different conditions and levels of detail. This tends to make the system robust in that an object which could not be located initially may be found later when the knowledge about the image has increased.

1. Knowledge-Directed Image Analysis

Image analysis is the process of attaching meaning to an image. One way to do this is to use an explicit model of what the image can contain, and then construct a mapping between the model and the image. Since a particular image is only an instance of the class of images that the model was intended for, the model must be in some sense "larger" than the image. That is, a useful model of the domain of an image will typically contain a large amount of information on possible image content. However, the mapping generated by the analysis process (if we have a good model), will typically use only a small portion of this model.

An even smaller portion of the model is used in specialized analyses, for example, analysis performed to look for particular objects in the image. These processes typically require that only a small portion of the image be mapped onto a small relevant part of the model that explains that image. For example, a request to analyze a radiograph for pneumoconiosis might use only a

small portion of a radiograph model. We term such a request for partial information a query, to emphasize that only a portion of a large possible mapping is generated. Figure 1.1 illustrates the concept of a query-directed mapping.

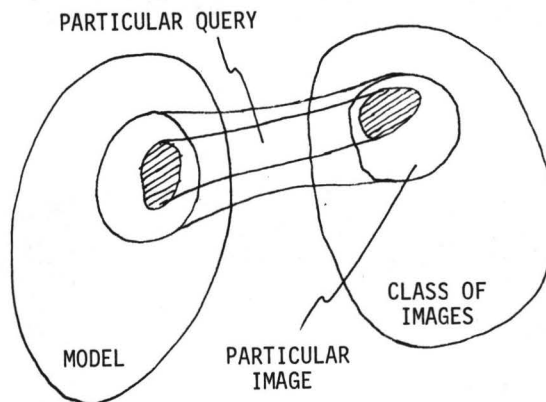


Figure 1.1 Query-Directed Mapping

The concept of a query is central to our approach to image analysis. Given a richly descriptive image model, a query can be coded in such a way as to require mapping a minimum of model structure onto the image.

Another aspect of our design is the retention and use of information gained in previous tasks so that mappings may be refined over a succession of queries. Examples of this would be:

- returning to an aerial photo on different days to perform different tasks;
- different physicians requesting different differential diagnoses of a radiograph;
- generating different land use maps for agricultural and social scientists from ERTS data.

For this approach to image analysis, it is important to define a representation that allows extensions of partial mappings which may be known a priori or acquired sequentially. Additionally, we must have a way of defining quantitatively when the query has been satisfied so that we do not perform unnecessary mappings (e.g., the system

The preparation of this paper was supported by the Alfred P. Sloan Foundation under Grant No. 74-12-5 and by the Defense Advanced Research Projects Agency under Grant No. F30602-77-C-0050.

may or may not need to know where all parts of an object are if it has "found" the object at some coarse level of detail).

One of the problems in generating the mapping to satisfy a query is that the mapping is between very different structures. The natural elements of the model are objects which are represented symbolically, whereas the natural elements of the image are "picture elements," or pixels. To bridge this gap, we have structured our vision system in layers as shown in Figure 1.2. At one end of the structure is a semantic network representing our model. The model contains generic information encoded as idealized prototypes of structures from low level (such as edges) to high level (such as complex assemblages of objects in the world). In the middle we have a sketchmap. This is a data structure that is synthesized during image analysis and provides associations between the model and the image. Like the model, the synthesized sketchmap is a network of nodes which, if the analysis is successful, turn out to be instantiations of a subset of model nodes. Additionally, sketchmap nodes contain associations with image structures, such as edges and regions, specific to the particular image being analyzed.

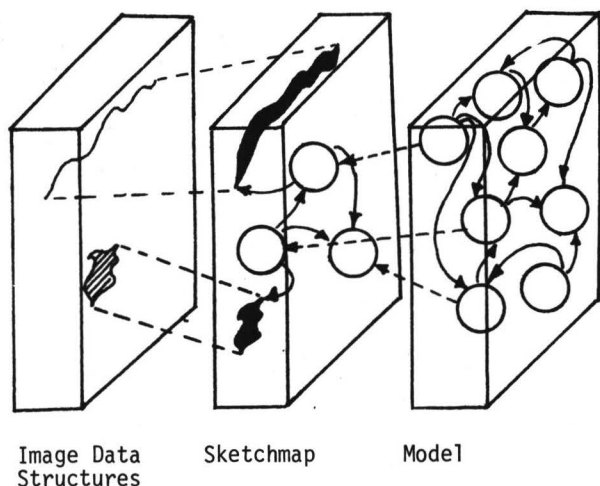


Figure 1.2 Basic Layer Structure

At the other end of the vision system is a third structure termed the image data structure. This consists of the original image at different magnifications, spectra, resolutions, etc., together with various filtered versions of the image, texture images, edge images, etc. By "edge images" we mean "primal sketch-like" images [Marr 1975]. The parameters for generating all the image data structures are typically specialized to a particular model-image mapping context.

A successful analysis of an image contains two parts: the generation of the proper links between sketchmap nodes and image data structures and the generation of the proper links between sketchmap nodes and model nodes. We describe a procedure that generates the image-sketchmap link

as a mapping procedure. The construction of links between the relational world model and the sketchmap is one of the functions of the executive procedure. The executive procedure embodies the overall strategy for achieving the goal(s) and is programmed in a high level language (currently SAIL). The problem of producing executive procedures from non-procedural specifications is not addressed in this project.

Since the best procedures for finding structures in real-world images are special purpose, we have decided to hand-code the mapping procedures in languages which produce efficient code. However, program autonomy is allowed in the choice of mapping procedures. Where different mapping procedures can generate the same links between sketchmap nodes and image data structures, the executive procedure can select the most appropriate based on a description attached to each mapping procedure. Also, the executive procedure can use general geometric relational constraints to pin down the location of objects.

2. The Structure of the Model

2.1 The Concept of Template Nodes

The model holds different kinds of knowledge about the image domain. It includes a relational network of nodes which are identifiable with (primitive and complex) objects and concepts in the domain from which the scene is taken. The answer to a query is a synthesized sketchmap or description; this is an instantiation of a subset of the model. The model, therefore, contains knowledge in the form of all potential instantiable descriptions. An example of this kind of knowledge is the assertion:

"the sternum is above the heart."

This can be readily included in the model network. It is potentially part of the model-image mapping ("the sternum" and "the heart" could be instantiated with pointers to regions of the picture). The model also contains knowledge which is not in the form of a synthesized description, but which, for instance, could be used in generating a description. An example of this kind of knowledge is:

"ships are about 6 times as long as they are wide."

This knowledge can be included in the model network and may help a program that is searching for a ship, but will not become part of the model-image mapping. When it is meaningful to differentiate between these two kinds of knowledge, we will refer to the parts of the network that represent the former kind as template nodes. Synthesized descriptions will be directly related to these nodes and their arcs.

Each template node has a substructure which represents the sense in which that node is to be "understood." Prior to a query, the meaning of a node is defined in terms of a substructure of mapping procedures and constraint relations. After

a query has been satisfied, the template node is represented by instantiated nodes with attached specialized location descriptions as shown in Figure 2.1. Four basic types of links provide a simple syntax to the network structure. A powerful advantage to this syntax is that the executive procedure can direct the analysis in a more general way.

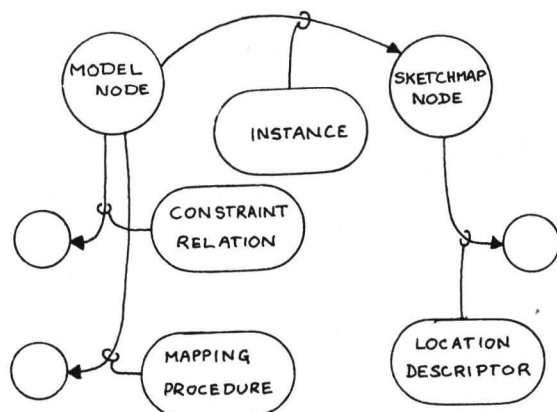


Figure 2.1 The Next Level of Detail in Model-Sketchmap Nodes

2.2 Constraints

Links to other model nodes encode (perhaps parametrized) constraint relations between model nodes. Links can encode:

- the probability that the relationship holds;
- a quantifier representing the expected value of the relationship;
- conditions under which the relationship holds.

For example, the relationship SHIP ADJACENT DOCK might have a certain probability of being true, an expected distance that the ship is from the dock, and conditions for the relation being true. We refer to the template nodes and geometric relations between them as the constraint network. This network may be interpreted like a program to find subsets of the model or the image that satisfy the constraints. Its results take account of partial or unspecified information, and it may be updated upon receipt of better data with a minimal amount of work. It is much like the graph of variable dependencies in AL [Feldman et al. 1975]. In brief, each node has a "Constraint Operation," such as Intersection, Translation, Union, or indeed any function of up to two arguments; it has two operand nodes; a father node; a status that may be "Up-To-Date" or "Out-Of-Date"; and a value that is some data structure such as a number, a list of linear objects, a region, etc.

The constraint network for the prose: "The centroids of docked ships are on lines parallel to the intersection of coastlines with dock areas at a distance of one-half a ship width" is shown

in Figure 2.2.

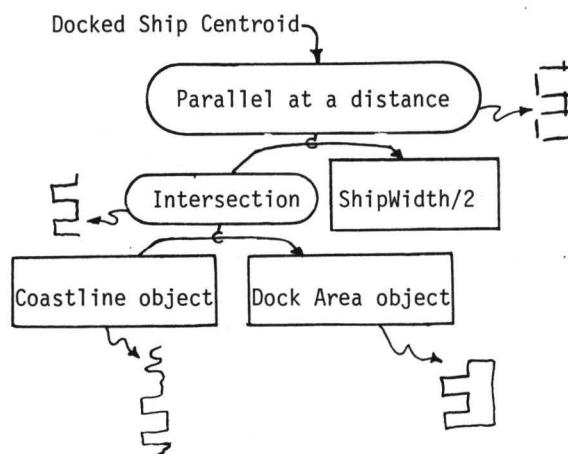


Figure 2.2 A Portion of a Constraint Network

The network starts out with data (from the model or from previous scene analysis) as the values of the tip nodes, but no values at nonterminal nodes and all nonterminals marked Out-Of-Date. Data at a tip node can have one of three statuses: it can be known that the object does not exist in the scene (so the value of the node is the null set), it can be known to some degree of accuracy where objects are in the scene (so the value of the node is a subset of image or world points), or perhaps nothing is known (in which case the object could be anywhere, and the value is implicitly the universe of image or world points).

When the constraint network is "run" to determine what is known about the location of its object, each node recursively evaluates its Out-Of-Date operand nodes, performs its operation, and stores the result in its value. It marks its status Up-To-Date. Intersection and Union work properly with the definitions of partial information of the last paragraph. When new (or better) information about an object at a tip of the graph comes in, all nodes on a path from the tip to the root are marked Out-Of-Date. Then when the graph is next run, (only) the necessary partial results are re-computed. In keeping with our philosophy, the graph is not self-activating, but is run on explicit user command.

2.3 Location Descriptors

A location descriptor provides information about where to find an entity. The part of the location descriptor which specifies a point set enclosing the region has been referred to as a tolerance region [Bolles 1975]. A shape location descriptor might have the structure shown in Figure 2.3.

This organization is suggestive of a frame-like structure. However, not all the entries need exist; just the syntax is necessary to allow the entries to be found. In practice only the

properties relevant to a particular query will be generated. Such partial instantiations are easy in our implementation language LEAP [Feldman & Rovner 1969].

```
[ ShapeLocationDescriptor
  nodetype: specialization prototype
  instance-of: a LocationDescriptor
  locates:    OneOf {(a ShapeObject),
                    (a ShapeFeature)}
  coordsystem: a CoordinateSystem
  centroid:    a PointSet
  //allows for "fuzziness"
  orientation: an AngleRange
  //...ditto
  tol. region: a PointSet ]

...similarly for Point, Linear, and
AreaLocationDescriptors

[ CoordinateSystem
  nodetype: abstract prototype
  units:    a LengthUnitSpecification
  scale:    a NumberRange
  //length units / system unit
  transforms: SetOf {(a Coordinate Transform)
                    (a Coordinate System)),
                    ...} ]
```

Figure 2.3 Example of a Shape Location Descriptor

There are many advantages to having a standard representation for object locations:

- If such descriptions are data types, their computations can be separated from the procedures that use them. If they can be passed as arguments, they provide a certain "common currency" between procedures, thus simplifying and modularizing the procedures that use them.
- Location descriptors can represent approximate locations, which is useful for queries unconcerned with exact answers.
- Constraints between locations can propagate knowledge throughout the model. Location descriptors can be computed from other location descriptors via relations, or by union and intersection of the described point sets. A system which applied linear programming techniques to the problem of locating regions through constraints placed on their boundaries was developed in [Taylor 1976].
- Use of location descriptors is geared to an abandonment of the exhaustive segmentation paradigm wherein every region must correspond to some object. Different location descriptors may refer to disjoint point sets or may overlap on the image, and different objects may have similar location descriptors.

3. Control

3.1 General Philosophy

Generally a query results in the synthesis of a sketchmap with instance nodes whose location descriptors are accurate enough for the purposes of the query. A query might also result in further refinement of location descriptors of the extension of an existing sketchmap to account for more image structure. A query-directed vision system should thus be able to use relevant information generated in successive queries. Most queries will take the form of user-written programs, since nontrivial tasks usually require fairly rigid recommendations about how the system should go about solving them. Initially the system will not attack the problem of automatically translating queries in some command language into programs. At the highest level control is embedded in the form of a user-written executive program. Figure 3.1 shows a fragment of the code used in the current executive procedure for selecting mapping procedures which identify instances of rib nodes in chest radiograph images. Each mapping procedure has pre-conditions, including an associated accuracy measure, which can depend on its neighbors, as well as a cost measure. The cheapest rib procedure which satisfies the pre-conditions is selected. The important point here is that the executive can have a relatively simple structure. This facilitates the experimentation of various control strategies other than the depth-first strategy shown in the example.

```
Recursive Procedure MatchRib(itemvar Node);
begin
  itemvar x,v; integer Var;
  if INSTANCE of Node is ANY

  then
    begin
      Print("rib ",Node," already matched");
      return;
    end
  else
    ! run procedure that will do the job with min cost;
    begin
      itemvar TempProc; integer MinCost,TempCost;
      MinCost := VeryLarge;
      foreach x such that
        RIB!PROCEDURE of Node is x do
        begin
          Var := GetConstraintsAndVariance(Node,x);
          if Var < Tolerance then
            begin
              TempCost := FINDCost(Node,x);
              if TempCost < MinCost then
                begin
                  TempCost := x;
                  MinCost := TempCost;
                end;
            end;
          end;
        end;
      if MinCost = VeryLarge
      then
        Print("No procedure can do job for ",Node)
      else ApplyProc(TempProc, Node);
      foreach v such that NEIGHBOR of Node is v
        and TYPE of v is RIB do
        MatchRib(v);
      end;
    end;
  end;
```

Figure 3.1 Executive Procedure for Ribs

3.2 Characterizations of Mapping Procedures

Mapping procedures have associated descriptions which are used by executive procedures. The descriptions contain the following:

- the slots in the data object which must be filled for the procedure to run;
- the slots the procedure can fill in;
- the cost and accuracy of the procedure in some meaningful units;
- the a priori reliability of the procedure.

Examples of the kinds of facts we expect to be able to encode in this structure are (for a straight-line structure) that a Hough transform cannot find the endpoints of a line but is more reliable than the cheaper Shirai tracker, which itself needs to know the direction of a line before it can track it, and that a Heuckel operator is more expensive, but can furnish many facts about the line with little known a priori, and can rate itself on reliability of its result.

There are several advantages to separating the executive procedure from the mapping procedures and their descriptions:

- a. The executive procedure can be written without considering the implementation details of mapping procedures in great depth.
- b. Mapping procedures are similarly simplified without the burden of determining an appropriate context for their application.
- c. The executive procedure can automatically select alternative procedures in the event of mapping procedure failures.
- d. Descriptions allow a choice between methods (if several are available) based on capability, resource requirements, and a priori reliability. (Also, recovery from failure of individual routines can be automated through planning [Feldman and Sproull 1975].)
- e. If the mapping procedures can produce reliable a priori estimates of their success, the analytical results of [Bolles 1975] and [Taylor 1976] could be extended to select the procedure which produces sufficiently exact data objects.

4. Applications

4.1 Finding Docked Ships

Finding ships in a dock scene illustrates how high-level metrical knowledge about the image (such as provided by a topographic map) can make certain scene analysis problems easy.

The model contains in a Constraint Graph form (see Section 2.2) the knowledge that docked ships are in the ocean adjacent to dock areas, parallel to the dock and with centroid a distance away related to the width of the ship. In a Shape Object Descriptor, some facts about the sorts of ships we are trying to find are stored,

viz. a template for matching them (in our case, a rectangle of 1's in an array for template-matching), their width, length, average brightness, etc. Template-matching is among the simplest vision primitives. Only in a context having a great deal of structure could it be expected to work in scenes as complex as Figure 4.1.

The system, under direction of the user-written query, begins by deciding where to look by satisfying a constraint network; the more information provided, the narrower the focus of attention. In the case illustrated in this section, the constraint network looked as it does in Section 2.2. Figure 4.1 shows a halftoned version of the image, with the dock area and coastline overlaid as black lines. Recent work at SRI [Barrow and Tenenbaum 1977] has shown that map data may be automatically registered with images such as ours to within better than a pixel, so we felt comfortable about bypassing the registration problem in this study. Were the registration uncertain, the constraints would produce a more fuzzy area to search than they did. Shown in the ocean are black lines indicating the areas of search which arise from the constraint tree; their linear nature makes a simple template-matching technique a possibility (in this exercise it was the only technique, but an executive procedure might well have chosen it as applicable). The ship template is rotated to be parallel to the midline as given by the constraint graph, and template-matching is done along the line; note is taken of where the score for the match goes over threshold, and when it comes back down under threshold. The average of these two positions is taken as the location of a ship. The black crosses show the results.

Our image data is a USGS mapping photograph provided by SRI, digitized to 256 grey levels on a .007" grid at USC. The image is stored on disk in RV format, and can be sampled at integral size reductions into an integer array in core for processing (see [Maleson & Rashid 1977] re. RV format).

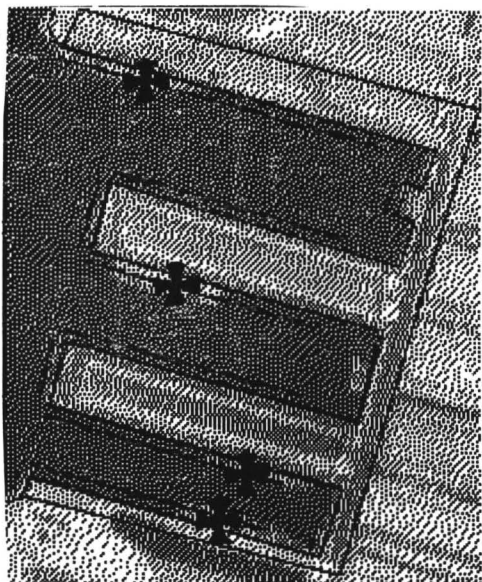
Linear objects are SAIL records making linked lists of (x,y) points. They can have four types at present: a list of points to be connected in order; a list of segments, or pairs of endpoints to be connected pairwise; and logically circular lists of points representing boundaries. A robust and general routine based on merging was written to compute the intersection of such linear features. Other useful geometric routines find the distance of a point from a segment (not a line), and compute a segment parallel to and some distance from another segment.

Regions (except for templates, which are arrays) are SAIL list items. A region is a list of y-lists; a y-list has a y-value followed by an even number of x-values. The first x-value is an "entering region" boundary point, the second is a "leaving region" boundary point, and so on alternately. The region: 001

101
011

would be represented as ((1 2 3)(2 1 1 3 3)(3 3 3)). Routines were written, again based on merging, to create the union and intersection of such regions, and to convert (via an asymmetric DDA algorithm [Newman and Sproull 1973]) linear objects to regions. We find multiple representations of objects simplifies the work of routines such as the constraint primitives.

Template-matching utilities can produce an array containing a rotated and scaled version of a template and can compute the correlation of a template (at some rotation and translation) with the image array.



Ship1 at x= 74.00000 y= 23.50000
 Ship2 at x= 83.50000 y= 39.00000
 Ship3 at x= 59.50000 y= 102.5000
 Ship4 at x= 49.00000 y= 171.5000
 End of SAIL execution

Figure 4.1 Harbor Scene with Dock Areas, Lines of Search, and Detected Ships

4.2 Finding Ribs in Chest Radiographs

The problem of finding ribs in chest radiographs illustrates the use of multiple procedures attached to the same template node and the use of less precise geometric constraints arising from anatomy rather than cartography.

The model contains nine right and left ribs (the maximum amount normally visible on a chest film). Presently only the lower edge of each rib is detected. Each rib is modelled as a template node with offset parameters from itself to each immediate neighbor (above, below, opposite). Additionally, three kinds of procedures are attached to each rib node as shown in Table 4.1.

LookForARib uses the Weschler parabolic model [Weschler & Sklansky 1975] to find a rib segment.

AffirmARib translates that segment using the offset parameters and attempts to verify the presence of a rib by a correlation technique. HallucinateARib instantiates a rib by translating a neighbor with no verification.

Table 4.1 RibFinding Procedures

Procedure	Preconditions	Cost	Var.	Postconditions
LookForARib	none	20	0	instance of rib
AffirmARib	instance of neighbor in sketchmap	4	1	instance of rib
HallucinateARib	instance of neighbor in sketchmap	1	5	instance of rib

Figure 4.2 shows the result of finding ribs on a particular chest film. The ribs are labelled with special symbols to denote the procedures which located them.

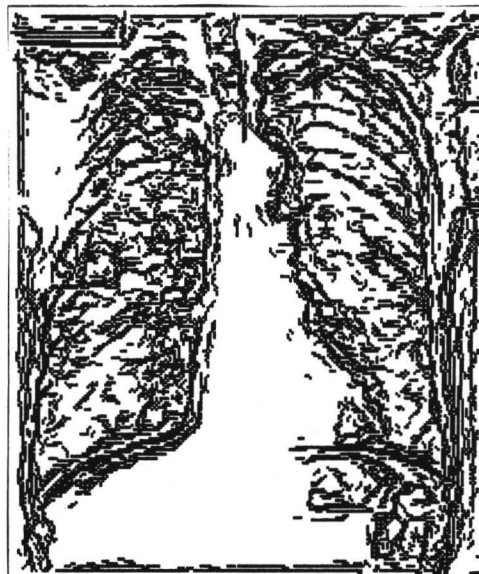
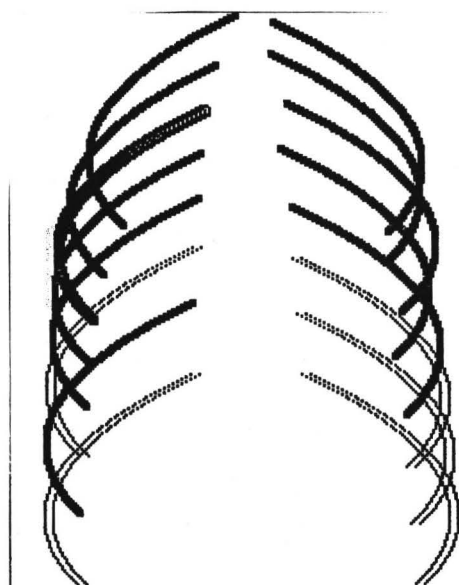


Figure 4.2a Local Edges Detected by Hueckel Edge Operator



Legend:  Found by LookForARib
 Found by AffirmARib
 Found by HallucinateARib

Figure 4.2b Results of Rib Finding

5. Summary

The semantic network is a kind of lumped parameter model in the spirit of [Fischler and Eschlager 1973]. The geometric constraints in the network relate template nodes whose descriptions (the "lumped parameters") are generated by attached mapping procedures. The key difference is that information found during the analysis can change the way template nodes are located.

In analyzing an image it is crucial that the generating of abstract descriptions of parts of the image, segmentation, be intimately connected with the interpretation of those parts. In our system the former operation corresponds to generating sketchmap-image links whereas the latter corresponds to generating model-sketchmap links. Interpretation and segmentation are united through multiple mapping procedures and the executive, which can efficiently change the way a part of the image is analyzed as new information about the rest of the image develops.

Finally, we want the image analysis process to do as little work as possible to satisfy a given task or query. This is attempted through the specialization of all parameters to the given task, the inclusion of performance and accuracy measures in the mapping procedure descriptions, and the use of the constraint network. All of this is just the beginning of a long term effort to

study what can be done in a general way for goal-directed image understanding tasks.

References

- Barrow, H.G., Tenenbaum, J.M., Bolles, R.C., and Wolf, H.C. "Parametric Correspondence and Chamfer Matching; Two New Techniques for Image Matching," DARPA Conference, Minneapolis, MN, April 1977.
- Bolles, R. "Verification Vision Within a Programmable Assembly System," Stanford AI Memo, AIM-275, December 1975.
- Feldman, J.A., Finkel, R., Taylor, R., Bolles, R., and Paul, R. "An Overview of AL, A Programming System for Automation," *Proc. 4th IJCAI*, Tbilisi, Russia, 1975.
- Feldman, J.A. and Rovner, P.D. "An Algol-Based Associative Language," *CACM*, Vol. 12, No. 8, pp. 439-449, August 1969.
- Feldman, J.A. and Sproull, R.F. "Decision Theory and Artificial Intelligence II: The Hungry Monkey," TR2, University of Rochester Computer Science Department, November 1975.
- Fischler, M.A. and Eschlager, R.A. "The Representation and Matching of Pictorial Patterns," *IEEE Trans. on Computers*, C-22, January 1973.
- Maleson, J. and Rashid, R. "The Rochester Image Protocol," University of Rochester Computer Science Department, Internal Memo, February 1977.
- Marr, D. "Analyzing Natural Images," M.I.T. AI Lab, AIM-334, June 1975.
- Newman, W.M. and Sproull, R.F. *Principles of Interactive Computer Graphics*, McGraw-Hill, Inc., 1973.
- Taylor, R.H. "Generating AI Programs from High-Level Task Descriptions," Ph.D. Thesis, Stanford AI Lab, 1976.
- Weschler, H. and Sklansky, J. "Automatic Detection of Rib Contours in Chest Radiographs," *Proc. 4th IJCAI*, Tbilisi, Russia, 1975.