VAS – A VISION ADVISOR SYSTEM COMBINING AGENTS AND OBJECTED-ORIENTED DATABASES

James L. Eilbert, William Lim, Jay Mendelsohn
Grumman Corporation/CRC
M/S A01-26
Bethpage, NY 11714-3580

AIAA-94-1181-CP

Ron Braun and Michael Yearwood
Grumman Corporation/CRC
M/S A04-12
Bethpage, NY 11714-3580

ABSTRACT

A model-based approach to identifying and finding the orientation of non-overlapping parts on a tray has been developed. The part models contain both exact and fuzzy descriptions of part features, and are stored in an object-oriented database. Full identification of the parts involves several interacting tasks each of which is handled by a distinct agent. Using fuzzy information stored in the model allowed part features that were essentially at the noise level to be extracted and used for identification. This was done by focusing attention on the portion of the part where the feature must be found if the current hypothesis of the part ID is correct. In going from one set of parts to another the only thing that needs to be changed is the database of part models. This work is part of an effort in developing a Vision Advisor System (VAS) that combines agents and objected-oriented databases.

INTRODUCTION

The bulkheads of Grumman aircraft, including the E2C, are assembled using a visually guided robot cell, called the Flexible Assembly System (FAS). Parts are laid out on a tray with the surface of the part that is to be attached to the bulkhead against the tray. Each part has a flange that is perpendicular to the tray. The robot receives information about the position and orientation of parts on the tray from a 2-D vision system which looks directly down on the tray, located about 6 feet away from the cameras. This means that the flanges that the vision system must locate are viewed edge on. FAS uses the coordinates supplied by the 2-D system to move a robot arm to the designated pickup point on a part which is always located on the flange. Once the arm is in position, the part is picked up at the pickup point. A 3-D camera with very limited range is used to find the positional error between a marker hole on the part and a reference hole on the gripper. Correcting this error allows the robot to determine the position of the part on the gripper accurately for placement on the bulkhead. After the part is placed against the bulkhead, the robot rivets it in place.

The Vision Advisory System (VAS) reported in this paper concerns the identification of parts and the location of their flanges using tray images such as the one shown in Fig. 1. Our goal is to make VAS an autonomous visual recognition system where the only change needed when the robot begins work on a new part set is a database describing the new parts. VAS is currently in the evaluation stage. It runs on a Macintosh 2fx connected via NFS to a Sun computer which runs the old 2-D vision system. Thus, it operates on the existing FAS manufacturing system, in parallel to the older, less-than-satisfactory 2-D vision system. The current

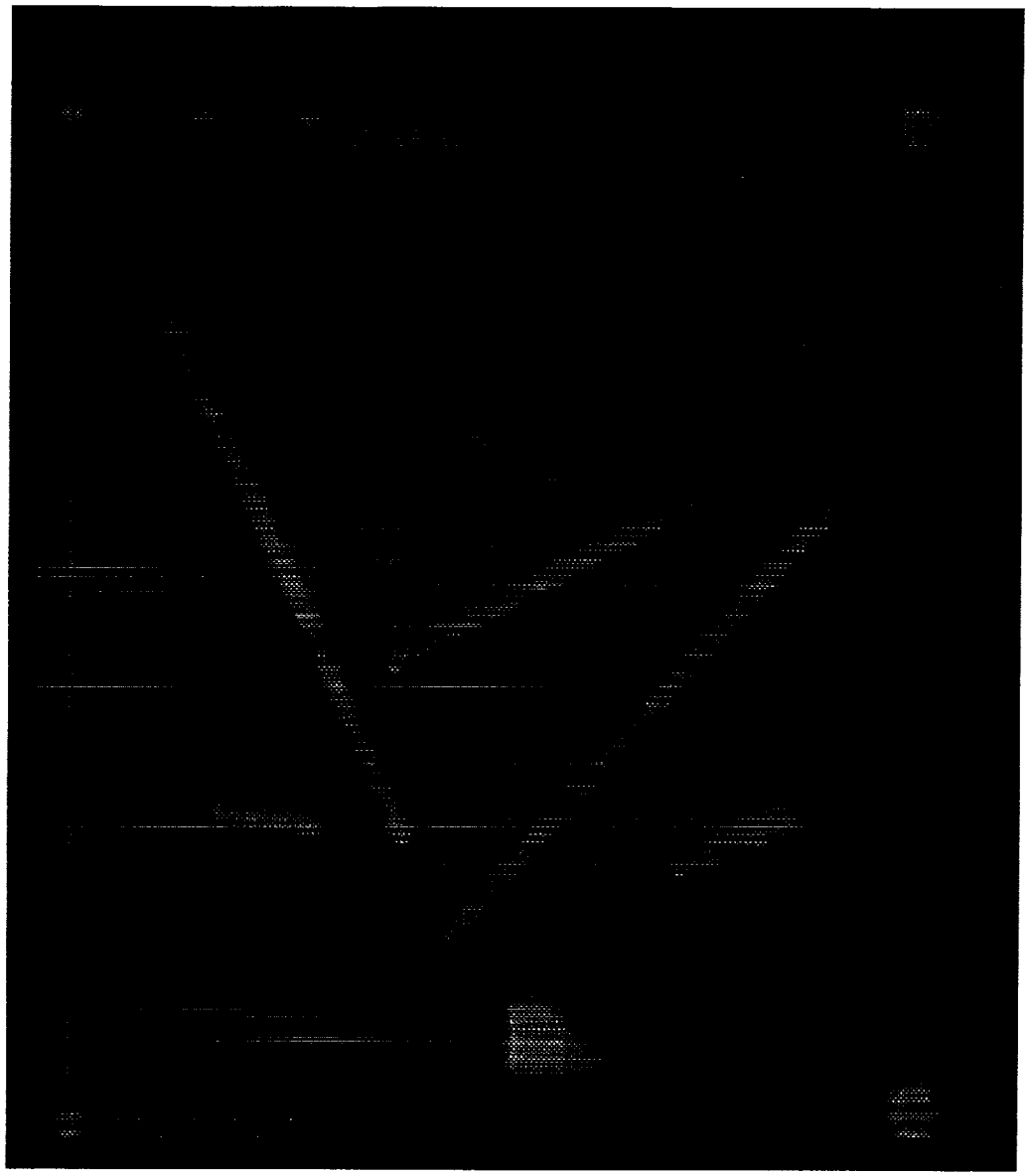

Fig. 1. A Typical tray image in which the fiducial marks in the corners were found and used to calibrate distance.

role of VAS is to provide "advice" regarding part identification and flange location to the existing system so a better decision can be made with regard to where the part pickup point is.

SYSTEM CONSTRAINTS

Although the parts never overlap or even touch one another the problem being addressed is made difficult by the similarity of some parts. Often the only difference between parts is the position of *features* on the sides of the parts, such as *bumps* and *notches*. These features can be viewed as convex or concave imperfections in the normally smooth and straight sides of the parts. Since the relative positions of the flange and the part's features are stored in a database, finding a feature solve the flange location task as well as the part ID task. The previous 2-D vision system, built in the mid-1980s, used a coarse shape description that generally ignored these small features. In

fact, it is the inability of that system to deal with small features that lead to the the current upgrade.

Since the robot was already in production when our task began, we were constrained to use the components of the existing system. As a result, the features (i.e. bumps or notches) that can distinguish parts or indicate the flange location are only 1-2 pixels in height or depth. Due to their small size, the precise position and extent of features on the parts cannot be determined.

In addition to the small features there are two previously unused sources of information that could simplify the solution to the 2-D recognition problem, namely shadows and context information. In cases where there is a clear shadow, it is possible to find the flange on a part even if it has no features at all. However, due to assymmetries in the lighting, shadows that are reliable predictors of the flange in one orientation cannot be seen at all in others.

Context information, i.e. the part descriptions in the database and the history of the current assembly session, can be used to augment and simplify the recognition process. For example, the knowledge of what parts have already been removed from the tray, can allow a part to be trivially identified if all of the parts similar to it have already been removed. The search for a feature in a small region under the assumption that some candidate model corresponds to the true part is also an example of the use of context information.

Since rule-based reasoning is relatively expensive in terms of the time it takes, the system avoids reasoning about context when possible. In cases where a sequence of inexpensive image operators leads to unambiguous results, the system does not do any additional reasoning. However, when there is ambiguity the system is able to reason about a part's ID or flange location using context information or even to decide that additional information must be extracted from the tray image.

The goal of only switching the physical descriptions of the parts making up the data set is not possible unless the system has all the image operators it will ever need. In particular, it assumes that image operators exist which allow any two features that can be found on any part to be distinguished. This is a not possible when you do not know what the parts in future sets will look like. To deal with this type of novelty the system must be able to "learn" to descriminate the new features from all existing features. In order to meet these requirements the system we propose must be able to do a limited amount of planning, learning, and high level representation.

A PROPOSED VISION ARCHITECTURE COMBINING AGENTS AND OBJECT-ORIENTED DATABASES

Following Minsky's [15] Society of Mind paradigm, researchers in a number of fields have begun proposing agent architectures. The emerging interest in Distributed AI [14,11,10,18] and in distributed control systems [5] has literally forced researchers to look at agent architectures of various types. However, researchers looking at autonomous systems that have multiple goals or drives and operate in several domains have been equally drawn to agents [1,2,3]. The solution to the FAS 2-D vision problem discussed above requires an autonomous system carrying several tasks with several domains in which it must be knowledgable (i.e. tray images, databases, robot arm coordinates). This suggested that the 2-D vision system could be naturally implemented as an agent architecture with a set of autonomous agents interacting with each other and an object-oriented database. In fact, the agent architecture chosen is a simplified version of an architecture originally developed for Automatic Target Recognition tasks [6]. In this paper, we describe the building blocks being implemented to support such an architecture. For example the agents and the object-oriented database are implemented in CLOS (LISP), while the basic image operators are written in C. Note that at present the agents we have implemented do not have the full capability of the agents

we envision, since the full support structure for the agent architecture is still being implemented.

Agents

The particular agent architecture used was developed at Grumman (13), and is an integration of the object-oriented programming paradigm and expert systems. Agents are both local experts and objects. Two basic classes of agents are available in the architecture: agents that manage a behavior and agents that plan to satisfy a goal or drive using a sequence of behaviors. Since the concepts of behaviors and plans play a central role in our approach and the terms are used in such a wide variety of ways, we provide the following list of definitions.

- *Behaviors* are a triple of actions, i.e. {activation, execution, termination} .
- *Routines* are control algorithms or a sequence of mappings between sensations that correspond to the execution portion of a behavior.
- A *domain* is a set of similar environments (ex. trays 1-5 of the FAS robot cell). In general, domain is a set of environments that is "sufficiently similar" to a prototype or a set of examplars, where both the prototype and the measure of similarity must be included in the definition. A routine may have a domain of validity associated with it.
- A *landmark state* is a description of the relation between the robot and important objects in the world.
- A *plan* is triple of events {recognition of start state, plan execution, recognition of goal state}.
- An *intention* is a sequence of mappings between landmark states that correspond to the execution portion of a plan.
- A *plan domain* is a set of similar situations (ex. part in reach on any of trays 1-5 of the FAS robot cell). Again a meaningful definition requires a prototype or examplars, and a measure of similarity must be included in the definition.

The behavior-managing agents are concerned with the moment to moment interaction between an entity and its environment, and the interpretation of sensation. The behavior-managing agent stores a set of routines plus information about its domain of applicability. In addition, it must be able to receive and store information about starting and stopping states from the planning agents with which it communicates. It must be able to translate these states into predictions about the corresponding sensations which it will actually detect. Behaviors have been developed for finding part boundaries, long-lines on the boundaries, and the bounding rectangle; and for detecting bumps, dents, and shadows. A behavior managing agent for "bump detection in the middle of a part" would decide when and where the search should take place, as well as when the search has succeeded and when it has failed.

The planning agents are concerned with "landmark states" and how to move between them. Each step in a plan must correspond to the resulting state change that occurs when a behavior is executed (12). Planners are incapable of operating in "real-time" since they do not have access to the real world through sensations. However, they may know what sequence of landmark states they will pass through before they need to stop planning. The planner stores a set of intentions plus information about its planning domain. In addition, it must be able to receive and store information about the current states from the planning agents with which it communicates. It must be able to compare these states with expected states to determine if the plan is working. A planning agent whose intention is to "locate flanges", would decide what combination of shadows and features to use in finding the flange and how to weight them. It would also send activation and termination states to the appropriate managing agent.

Agents combat the traditional brittleness of expert systems, associated with operating in too large a domain, by having many task specific behavior-managing agents that are competent in small domains and much fewer planning agents that monitor their applicability and performance. Like

other objects, agents can communicate by passing messages and they also have dedicated communication lines to other agents. There are also communication lines to the objects in the object-oriented database.

The Object-Oriented Database

Model-based vision requires data bases to store both models and various types of information obtained from the actual images. Most of researchers who have looked at the image database problem have advocated using object-oriented techniques even when their specifications are significantly different [8]. The object-oriented database utilized for VAS must store two type of information, in addition to the description of parts (in terms of sensation): spatial relationship of parts on the tray over time, and the plans that have successfully been used to do the major tasks. We have described elsewhere a high level representation consisting of the grid map, a graph of landmark states and a set of part descriptions that can organize this type of information [6]. The grid map describes a particular environment and the spatial relationship of objects within it. In this case, the environment is a tray and the the relationships are among the parts, fiducial marks, and clutter. The grid map is a bird's eye view constructed from a set of scenes that shows the relative positions of the important objects, but little detail of their internal structure. The graph of landmark states describes the plans that are valid in a given environment. The landmark graph is a network of (state) objects as is a standard AI semantic net [16]. However, the nodes of the landmark graph are connected to each other by plans for moving between states, rather than "ISA, PARTOF, or PROPERTYOF" links. Note that not all state nodes in the landmark graph involve "physical landmarks", some nodes involve temporary objects and are labelled as such. These two maps capture the spatial relationships and the plans learned for moving around an environment, and have most of the properties attributed to cognitive maps in living animals [17,7].

Discrimnation Net

Discrimination nets are a simple AI technique for classifying objects based on a set of common properties with two or a small number of values (4). The use of fuzzy properties to describe parts makes it possible to use a discrimination net to classify the part models in a database. When parts with very similar sizes and shapes must be identified, it is important to keep all reasonable candidates until a final discrimination is made. The discrimination net does exactly this. To use the discrimination net one would make a list of the properties of an image-object and run them through the net. Each property is used to pick a direction in the net until a leaf node is reached and a part ID is returned. If a leaf node is not reached a small number of candidates are returned. The discrimination net actually consists of a sequence of keys and linked lists. If the list (SHORT MEDIUM ((BUMP .1) (NOTCH . 0))) were submitted to the net, the relevant part of which is illustrated in Fig. 2, then both part 787 and 7101 would be returned. The decision of which of these parts is being examined would require looking at rough feature position or the quantitative measures of length or width. The issue of setting model-based matching criteria is a difficult problem in general (9), but our images have simple backgrounds and good part background separation which simplifies things.

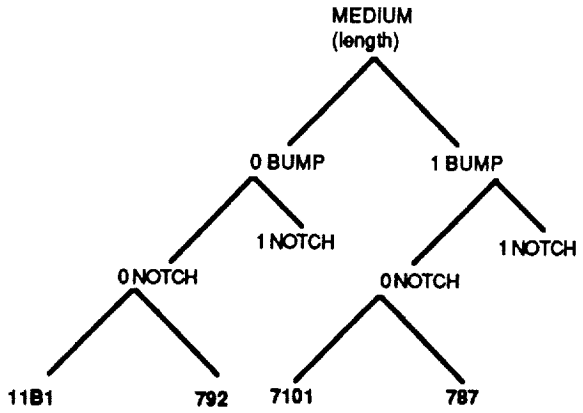

Fig. 2. A sample portion of a discrimination net.

THE CURRENT SYSTEM

There are six major tasks for the 2-D vision system, i.e. database completion, part outlining, long-line finding, part ID, flange location, and the sending of pickup coordinates to the robot arm. These goals each have a planning agent associated with them. Together these agents through their interactions carry the part ID and flange location tasks that is the real purpose of the 2-D vision system. Overall processing is initiated by the operator with a Lisp function called "run-FAS" which takes a database of part-models as an argument. The system then runs till the following midnight when it reports its results.

When the system first encounters a new data set the database completion agent is activated, it moves through each part model and extracts fuzzy descriptions of the length, width, and features. This information is stored in appropriate slots of the part models. The following is a typical part model description from the database where the slots in **bold** are automatically filled in by database completion agent:

```
;;; Part 715
(setq 715
(make-instance 'part-model
        :model-name '715
        :home-tray nil
        :home-bank nil
        :surface-list nil
        :center-of-mass nil
        :part-length 23.20
        :fuzzy-length nil
        :part-width 0.631
        :len-wid-ratio nil
        :fuzzy-width nil
        :bump-list '((0.0 .625 .128)
                     (6.75 7.39 .14)
                     (13.38 14.06 .14)
                     (20.63 21.49 .14))
        :hook-list nil
        :needle-list nil
        :notch-list nil
        :tail-list nil
        :easy-features nil
        :grip nil
        :flange-height 0.638
        :major-flange-bumps nil
        :flange-shape 'L
        :similar-part-list nil
        :action-list nil
        :action-code nil
        :group *load-group*))
```

This agent then builds a discrimination net for all parts in the data set and stores them in the experiment data object. When the database is complete, it sends a message which activates the part outlining agent.

The part outlining agent then monitors a working image directory to see if a tray image has been captured. It takes a pair of images to cover the entire tray. The basic algorithm that the part outlining agent uses consists of adaptive thresholding, morphological smoothing, and a boundary following procedure. The results of this process are shown in Fig. 3. When the part outlining agent completes its task it activates the long-line finding agent.

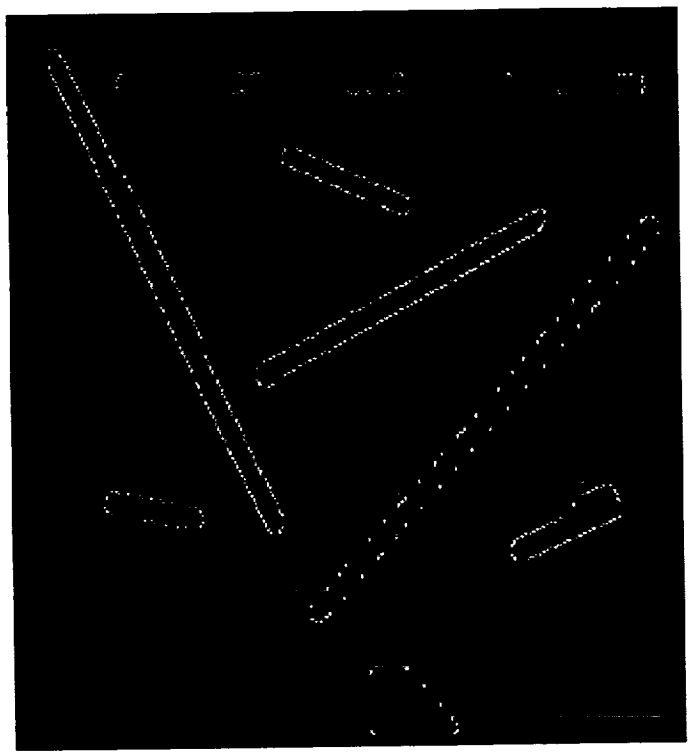

Fig. 3. Results of the Part Outlining Task

All of the parts with which we have worked are long and thin. If parts that do not have this general shape appear in the database a new agent that finds their bounding rectangle will have to be developed. The long-line finding agent uses a simplified Hough transform to find candidate line segments in the outline that may belong to the long lines. It then uses a mean square error best fit line on the line segments that are sufficiently similar. Fig. 4 shows the long-lines found for each part, plus the bounding rectangle for the long part on the left. If the long-line agent completes its task, it activates the part ID agent. Note that all of the processing up to this point runs automatically with only basic checks for failure. All of the information is stored in a data object called a tray. A partial listing of a filled tray follows:

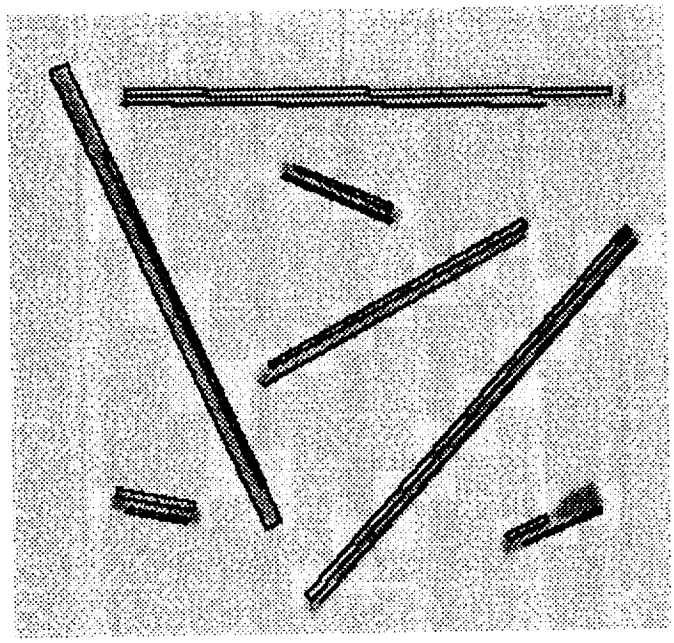

Figure 4: The Long-lines Found for the Parts (Color reversed for clearer graphic display.)

```
#D(TRAY
  TRAY-NAME
"Images:Shading:N_shortN.8bits"
  IMAGE-OBJS
  (#562=#D(IMAGE-OBJ
        PART-NAME NIL
        TRAY-NAME
           "Images:Shading:N_shortN.8bits"
        TRAY-COORD-CENTER (199 . 294)
        PART-AXIS NIL
        PART-LENGTH 252.906525
        PART-WIDTH 19.008202
        LEN-WID-RATIO NIL
        TRAY-COORD-ANGLE 0.0
        BUMP-COUNT 0
        BUMP-LIST ((NIL NIL NIL)
                   (NIL NIL NIL))
        NOTCH-COUNT 4
        NOTCH-LIST ((NIL (((312 . 287)
                           (313 . 288)
                    1.756594313390609))
                   (((89 . 286) (88 . 288)
                    3.1204771710092984)))
                ((((138 . 303) (129 . 303)
                  1.30427164452175))
                 NIL
                 (((125 . 303) (88 . 300)
                  4.814981468417682))))
```

```
ZOOMABLE NIL
INSIDE-INTENSITY-AVE
            ((73.0 11.346012 1139)
             (85.0 11.083988 910))
INSIDE-GRAD-AVE NIL
OUTSIDE-INTENSITY-AVE
            ((20.0 4.298134 769)
             (26.0 11.714762 994))
LONGEST-LINES (#D(ANGLINE
          ANGLE  0.00162448
          LN-LENGTH  222.001331
          Y-INTER NIL
          END1 (312 . 285)
          END2 (90 . 285)
          GROUP NIL
          )
        #D(ANGLINE
          ANGLE  -0.0124623417
          LN-LENGTH  187.010402
          Y-INTER NIL
          END1 (313 . 302)
          END2 (126 . 304)
          GROUP NIL
          ))
BOUNDING-RECT ((313 . 285)
          (313 . 302)
          (88 . 305)
          (88 . 285)) ...
```

The basic approach taken by the part ID agent is to use the fuzzy discrimination net describing the gross characteristics of all the parts in the current database. The part ID agent takes the information about a part which was found by outlining and long-line finding agents and fills in the fuzzy slots for its image part in the object-oriented

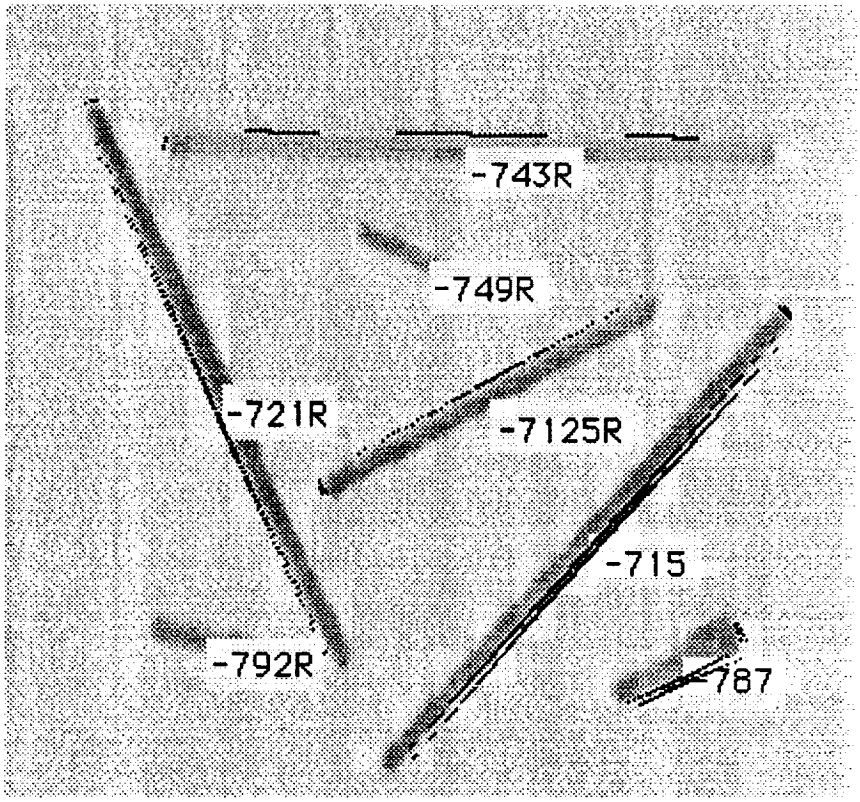

Figure 5: The Part IDs and the Flanges Found (Color reversed for clearer graphic display.)

database. The fuzzy information from the part in the current tray image is turned into a list and run through the discrimination net. A short list of candidate parts is returned.

For example, consider getting the ID of a part centered at (252 . 320) on the tray. It has no bumps nor notches and its length 5.15 (SHORT) and width 0.75 (FAT). The

candidates parts with the correct feature list are:

((SHORT FAT ((BUMP . 0) (NOTCH . 0)) PART119)
(SHORT FAT ((BUMP . 0) (NOTCH . 0)) PART70)
(SHORT FAT ((BUMP . 0) (NOTCH . 0)) PART69)
(SHORT FAT ((BUMP . 0) (NOTCH . 0)) PART55)
(SHORT FAT ((BUMP . 0) (NOTCH . 0)) PART49)
(SHORT FAT ((BUMP . 0) (NOTCH . 0)) PART59)
(SHORT FAT ((BUMP . 0) (NOTCH . 0)) PART82)
(SHORT FAT ((BUMP . 0) (NOTCH . 0)) PART81))

The candidates for the object are reduced to PART55, PART70, and PART59. If a definite ID cannot be made based on differences in the length and width, then small features are sought in particular places. Finally, the presence of IDed image parts activates the flange finding agent.

Initially, the flange finding agent ignores the image part IDs and executes a shadow finding routine. By setting the criterion for finding a shadow high we can force all classifications to be correct or unkown. If shadowing does not yield an answer then small features are sought where they should be based on the part ID. Fig. 5 shows the final IDs and flanges found. All IDs are correct and five of seven flanges were found correctly. In the two cases where the flange was not located correctly, i.e. 749 and 792, VAS reported that it could not find the flange rather than making an error. Note that neither part had features, and that human observers were also unable to locate those flanges. Of the five flanges found, 743 was located based on its shadow, while the other four were located based on finding features.

Each of these agents have contingency plans that are implemented when basic algorithms fail in particular ways. For example, if the part IDer does not come up with a unique ID, it will send a request to the database to give it the lengths, widths and the approximate location of the features on each of its candidate models. Agents also communicate indirectly with each other through the tray and image-objects. Since everything of use to any of the other agents is recorded on these objects, agents do not need to know which agent calculated a piece of information in order to use it. Thus, an agent will always check the database to see if a piece of information that it needs is available before it tries to extract it directly, or sends a request to another agent. Controlling the communication among agents is one of the major challenges of agent architectures, and is still being studied for VAS.

CONCLUSIONS

A model-based approach which uses fuzzy descriptions of part features for classification and an object-oriented database of parts has been developed. A variety of image processing techniques have been combined to find the information needed to do identifcation and flange location, i.e. length, width, small bumps and dents, and shadows on the parts. It was possible to decompose the overall task into a set of modular tasks that interact and fail in specific ways. An agent architecture has been developed that takes advantage of the modularity in this multi-task and multi-environment domain. The success of a vision architecture initially developed for a wholely different application, i.e. automatic target recognition give us hope that the agent approach to autonomous vision problems is a general one.

One final point is that with better cameras and lighting many of the problems that proved very stuborn in VAS would never have come up. However, good design is hard to do when the scope of the problems the system will face are not known in advance.

REFERENCES

1- Agre, P., Chapman, D. (1987) Pengi: An implementation of a theory of activity. Proc 6th National Conf on Artificial Intelligence, AAAI-6.
2- Anderson, T.L., Donath, M. (1988) A computational structure for enforcing reactive behavior in a mobile robot.

SPIE V. 1007: Mobile Robots III. Boston, MA.
3- Brooks, R.A. (1986) A robust layered control systemfor a mobile robot. IEEE J. Robotics and Automation, RA-2. 14-23.
4- Charniak,E., Riesbeck, C.K. McDermott, D.V., & Meehan, J.R. (1987) Artificial Intelligence Programming. 2nd ed. Hillsdale, NJ: Lawrence Erlbaum Assoc.
5- Cui, X., Shin, K.G. (1991) Intelligent coordination of multiple systems with neural networks. IEEE Man, Sys, Cybernetic, 21(6):1488-97.
6- Eilbert, J.L., Mendelsohn, J. (1992) Organizing and using context information for ATR. pp. 303-312. Proc. Second Automatic Target Recognizer System & Technology Conf.
7- Gallistel, C.R. (1990) The Organization of Learning. Cambridge, MA: Bradford Book.
8- Goodman, A.M., Haralick, R.M., Shapiro, L.G. (1990) Knowledge-based computer vision- integrated programming language and data management system. IEEE Computer 22(12): 43-58.
9- Grimson W.E.L., Huttenlocher, D.P. (1991) On the verification of hypothesized matches in model-based recognition.
10- Hewitt, C., Inman, J. (1991) DAI betwixt and between: From intelligent agents to to open systems science. IEEE Man, Sys, Cybernetic, 21(6):1409-19.
11- Kaebling, L.P., Rosenschien, S. R. (1990) Action and planning in embedded agents. pp. 35-48. In: Maes, P., Designing Autonomous Agents. Cambridge, MA: Bradford Book.
12- Lim, W., Eilbert, J.L. (1990) Plan-Behavior Interaction in Autonomous Navigation. SPIE V. 1388: Mobile Robots V:464-475. Boston, MA.
13- Lim, W., Verzulli, J. (1990) SAL -- A language for developing an agent-based architecture for mobile robots. pp. 285-296. SPIE VII. 1831: Mobile Robots VII. Boston, MA.
14- Maes, P. (1990) Designing Autonomous Agents. Cambridge, MA: Bradford Book.
15- Minsky, M. (1986) The Society of Mind. NY: Simon & Schuster.
16- Nilsson ,N.J. (1980) Principles of Artificial Intelligence. Los Atlos, CA:Morgan Kauffmann.
17- Tolman, E.C. (1948) Cognitive maps in rats and men. Psychol. Rev. 55:189-208.
18- Wong, S.T.C. (1993) COSMO: A communication scheme for cooperative knowledge-based systems. IEEE Man, Sys, Cybernetic, 23(3):809-824.