An Architecture for Vision and Action *

R. James Firby, Roger E. Kahn,
Peter N. Prokopowicz and Michael J. Swain

Department of Computer Science
University of Chicago
1100 E. 58th St., Chicago, IL 60637
firby,kahn,peterp,swain@cs.uchicago.edu

Abstract

Vision systems that have successfully supported nontrivial tasks have invariably taken advantage of constraints derived from the task and environment to increase reliability and lower the complexity of perception. We propose that it is possible to build a *general purpose* vision system, that is, one that can support a wide variety of tasks, *and* take advantage of such constraints. The central idea within our proposed architecture is the *reactive skill.* Skills are concurrent control routines assembled at run time using instructions from a symbolic execution system. Visual modules are used as resources in the construction of these skills. Skills control the agent as continuous feedback loops but are constructed using discrete, symbolic instructions. The key to general-purpose vision is the ability to parametrize the primitive elements of the vision system and to compose visual and control routines in a variety of ways. We demonstrate the architecture in the context of an implemented example task of a robot collecting trash off a floor and depositing it in a garbage can.

1 Introduction

A longstanding goal in computer vision has been to develop *general vision* that can support a wide variety of goals, and inform the observer of the relevant ways in which the world does not meet its expectations [Marr, 1982; Tarr and Black, 1994; Brown, 1994]. On the other hand, it is advantageous for working systems to make as much use of the task and domain constraints as possible; vision systems that have successfully supported nontrivial tasks have invariably done so (see e.g. [Horswill, 1993] [Dickmanns and Graefe, 1988]). Our goal is to build a system that is able to take advantage of task and environmental constraints, and yet is flexible enough to respond to dangerous situations and failed expectations, and can be quickly reprogrammed to do a different task or work in a different environment. The key to doing so is to build re-usable components that can be assembled in a task and environment specific way.

We call our proposal the Animate Agent Architecture. In the Animate Agent Architecture, visual processing is encapsulated in modules called visual routines (after [Ullman, 1984]) that can be invoked as needed, and which are in turn composed of combinations of primitives called visual operators. At run-time, visual routines are paired with action routines designed to run in a tight servo loop with perceptual routines to meet the real-time constraints of the task. The resulting process is termed a *reactive skill* [Slack, 1990; Firby, 1994]. Skills are invoked and can be terminated by a higher level system, that has access to the agent's knowledge of the task and environment, and is therefore well-suited to selecting the appropriate set of control and perceptual routines. For this component, we use the RAP system [Firby, 1989a; 1994]. In contrast to the tight link between the control routines and visual routines, the RAP system communicates with the skills via a relatively low bandwidth message-passing interface; we call the messages *signals.* We illustrate the architecture in the context of a trash collection task - one of the tasks of the 1994 AAAI robot competition.

Our design seeks to take advantage of as many constraints as possible in any given situation and encode the constraints that change from situation to situation at a high level, where they are easily changed. These constraints allow us to sidestep many of the difficulties that increase the complexity and lower the accuracy of computer vision algorithms. For example, when looking for a trash can, the robot is able to avoid searching over all possible views (or, alternatively, using only features that are independent of perspective) because we have a good idea of the view from which the robot will see the trash can (assuming it is sitting upright on the floor). We are also able to limit the search to only locations in

*This work was supported in part by ONR contract N00014-93-1-0332, ARPA contract N00014-93-1-1394 and NSF Grant No. IRI-9210763-A01.

the scene consistent with an object sitting on the floor, and to combine the results of different algorithms to increase the speed and reliability of the search.

Within this framework, not all sensing is task-driven. For instance, a visual operator monitoring possibly approaching objects to warn of possible collisions (such as Nelson's directional divergence operator [Nelson, 1989]) can raise a signal if it senses motion that appears dangerous. If the taking avoidance action is of higher priority than the ongoing task, the RAP system will interrupt the ongoing task to respond to the danger. If the response is known by the RAP system not to conflict with the ongoing task, it is possible to run the processes in parallel. While the current implementation does not implement Nelson's algorithm, it does have equivalent signals raised by the sonar sensors.

The entire trash collection task is described below, after a description of the proposed architecture and the robot that accomplishes the task.

2 The Animate Agent Architecture

The overall system design for the Animate Agent Architecture consists of two components: the RAP reactive plan executor, and a control system based on reactive skills. The executor executes *sketchy plans,* that is, plans that leave many steps to be chosen as the actual situation unfolds. It makes these choices at run time when the true state of the world can be sensed. The output from the executor is a detailed set of instructions for configuring the skills that make use of the agent's sensors and effectors.

2.1 The RAP Execution System

The RAP system expands vague plan steps into detailed instructions at run time by choosing an appropriate method for the next step from a preexisting library. By waiting until run time, knowledge of the situation will be direct rather than predicted and much of the uncertainty and incompleteness plaguing the planner will have disappeared. However, some uncertainty will always remain and incorrect methods will sometimes be chosen. The RAP system copes with these problems by checking to make sure that each method achieves its intended goal and, if it doesn't, choosing another method and trying again.

In the RAP system a task is described by a Reactive Action Package (RAP) which is effectively a context sensitive program for carrying out the task. The RAP can also be thought of as describing a variety of plans for achieving the task in different situation.

The RAP system [Firby, 1987; 1989b] carries out tasks using the following algorithm. First, a task is selected for execution and if it represents a primitive action, it is executed directly, otherwise its corresponding RAP is looked up in the library. Next, that RAP'S check for success is used as a query to the situation description and, if satisfied, the task is considered complete and the next task can be run. However, if the task has not yet been satisfied, its method-applicability tests are checked and one of the methods with a satisfied test is selected. Finally, the subtasks of the chosen method are queued for execution in place of the task being executed, and that task is suspended until the chosen method is complete. When all subtasks in the method have been executed, the task is reactivated and its completion test is checked again. If all went well the completion condition will now be satisfied and execution can proceed to the next task. If not, method selection is repeated and another method is attempted.

2.2 Reactive Skills

The RAP system refines tasks into detailed discrete actions at run time. However, one of the primary lessons of both recent AI research into robot control [Agre and Chapman, 1987] and recent vision research into active perception [Bajcsy, 1988] is that actions must be dynamic processes tolerant of sensor error and changing surroundings: they must be reactive. It is not useful for a planning system to control a robot with steps like "move forward 10 feet" because the location of the robot may not be known exactly, the world model may be inaccurate, or an obstacle such as a person may suddenly appear, and so on. Instead the goal must be given relative to something that can be sensed in the environment, and flexibility must be built in to the action in order to avoid unanticipated undesirable states. Experience also shows that a fruitful way to think of such a control system is as a collection of concurrent, independent behaviors [Brooks, 1986; Slack, 1992]. Such behaviors can be enabled in sets that correspond to the notion of discrete "primitive steps" that will reliably carry out an action in the world over some period of time. The RAP system produces goal-directed behavior using this idea by refining abstract plan steps into a sequence of different configurations for a process-based control system.

The resulting interface between reactive execution and continuous control is illustrated in Figure 1. It consists of instructions to enable, disable, and set the parameters of individual routines. Thus, the RAP system simply configures the control system and cannot change the routines available or the attributes that connect them.

Control routines communicate with the RAP system by sending signals. Signals are typically low bandwidth messages such as "I've reached the position I was supposed to get to" or "I haven't made any progress in a while." Each routine will have a set of signals it sends under various circumstances. Since routines do not typically know the reason they have been enabled *(i.e.,* the goal enabling the routine is known to the RAP system but not the control system), the messages they send carry very little information when taken out of context. It

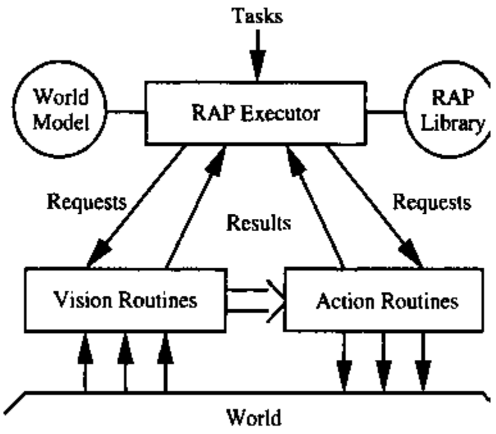

Figure 1: The Animate Agent Architecture

is up to the RAP system to interpret these messages in light of the goals it is pursuing and the routines it has enabled. For example, "I cannot move forward", might mean failure because there is an obstacle in the way if the robot is trying to cross a room; on the other hand, the same signal might mean success if the robot is supposed to move down a long hallway until it reaches the end. The task expansion structure built up by the RAP system during execution is ideal for this interpretation.

For example, suppose the next routine to be carried out is to move to a trash can. This routine can be implemented using the fairly generic action routine "move in a given direction while avoiding obstacles." The argument to this routine is the direction to move and that can be updated in real time by an active sensing routine that continually tracks the direction to the can. Possible routines might track the trash can's shape, color, or motion, depending on the situation. It is up to the RAP system to choose which is appropriate at run time. Once the action and tracking routines have been enabled, the control system will move the robot towards the can whatever gets in the way. This control state will last indefinitely without further intervention as long as the tracked feature stays in view and the path toward it stays reasonably clear.

2.3 Visual Routines

The visual system is based on a modular set of task-specific modules, termed *visual routines,* designed to sense the information required by the reactive skills. These modules are in turn composed of a sequence of primitive visual processing steps called visual operators. The operators and routines are designed with re-use in mind. The intermediate representations passed between operators are designed to be as generic as possible, so that they may be shared by multiple operators. With a common language of intermediate representations, it

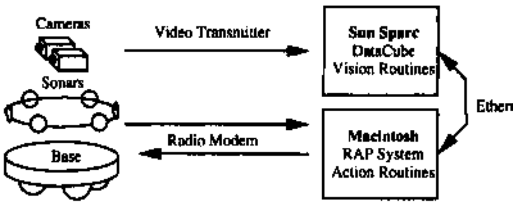
Cameras
Video Transmitter
Sonars
Base
Radio Modem
Sun Sparc DataCube Vision Routines
Macintosh RAP System Action Routines
Ethern

Figure 2: Schematic diagram of the robot

is posssible to compose many different routines with a much smaller number of primitive operators. While the routines are designed to be task-specific, as much as possible the task constraints are input as parameters, rather than being coded into the routine.

The base representation generated by the early visual system provided to the routines is a three-band (R,G,B) color Gaussian pyramid. Visual routines may access a rectangular subset of any level of the pyramid. All further representations are created on demand by visual operators.

3 CHIP

The robot we have constructed as a testbed for our ideas is shown schematically in Figure 2. It uses a three-wheeled omni-directional mobile base built by Real World Interface (RWI). Stereo color cameras are mounted on top of the robot on a two degree of freedom computer-controlled pan-tilt platform. Video signals are processed off-board by Datacube image processing hardware attached to a host Sun SparcStation. The robot sensors also include sonar sensors and infrared proximity sensors. The sonars can be turned to face the surface nearest the robot to minimize problems due to specular reflection. The infrared sensors are well equipped for detecting obstacles near the robot (their signal drops off quickly after about ten centimeters), and have failure modes that are complementary to the sonar sensors. For manipulation, CHIP is equipped with a Heathkit Hero arm and manipulator. Force and contact sensors on the gripper provide tactile feedback.

Microprocessors mounted on the robot run the sensors, do initial processing of the data they create, and skills that do not require visual processing, which is done off-board. The microprocessors share a small amount of memory to allow communication and synchronization. The on-board microprocessors talk via radio modem to a Macintosh computer, which runs some control routines and the reactive planner.

The robot can be run in either a tethered mode or a non-tethered mode in which the cameras transmit video to the Datacube and the Macintosh workstation communicates with the on-board microprocessors via a radio link.

The robot has enough freedom of movement to allow it to navigate through a considerable area within the Computer Science Department. Because people work within the robot's range of movement, the robot's world is always changing.

CHIP's vision processing is distributed over a Sun Sparc-20 workstation and a DataCube MV-200 image processing computer with a DigiColor digitizer. The DataCube is a pipelined processor that can perform many operations including convolutions and histogramming in real-time. The DataCube Server [Kahn, 1993] allows multiple processes to access the DataCube simultaneously from any machine on the network. The Server allows us to implement our visual routines as separate processes potentially distributed across multiple machines while still allowing each visual routine to access data from the DataCube.

There is a set of low-level visual operations that are widely used among the visual routines and/or are time-critical and so have been implemented on the DataCube. These operations include the creation of color Gaussian and Laplacian pyramids, extracting subregions from a level of one of the pyramids, convolution, and histogramming, color histogram backprojection and motion detection. Visual routines may access a rectangular subset of any level of the pyramid. All further representations are created on demand by visual operators.

During each visual routine's initialization a socket connection with the DataCube Server is established. When a visual routine requires a computation to be performed on the DataCube it sends requests to the DataCube Server. The DataCube Server performs the requested operation and returns the output. The programming interface to the DataCube Server makes all server requests look like normal function calls. Using this interface we are able to write visual routines that use the DataCube without worrying about its internals.

Visual routines communicate with the rest of the system through a mechanism called the *Message Hub* [Kahn *et a*/., 1994]. The message hub provides a reliable socket based communication link between processes with a simple programming interface.

4 Experiment: The Trash Collection Task

In one of the 1994 AAAI events, robots competed to tidy up an office area littered with soda cans, styrofoam cups, and paper wads, collecting and depositing them in any nearby trash can. Although CHIP did not win the event, it was the only robot to actually pick up a piece of trash and put it in a trash can. The RAP system made it easy to decompose the task into a set of simpler vision, navigation, and manipulation goals, and to add new context-dependent methods for achieving them. At the top level, CHIP's goal in this example is "put-trash-in-can". To an outside observer, the behavior looks like the sequence:

```
(sequence
  (tl (scan-for-object trash))
  (t2 (go-to-object trash 80))
  (t3 (orient-to-object trash 40))
  (t4 (pickup-object trash))
  (t5 (scan-for-object trash-can))
  (t6 (go-to-object trash-can 120))
  (t7 (orient-to-object trash-can 70))
  (t8 (drop-held-object))))))
```

Figure 3: Method for achieving the top-level goal *put-trash-tn-can,* The numbers indicate distances from the denoted objects (in cm).

find trash nearby — move to trash — align with trash — pick up trash — find trash can — travel to trash can — align with trash can — dump trash in can — turn back toward where trash was found — repeat

The method for achieving this goal is the sequence of subgoals in Figure 3. All these goals lead to further subgoals; ultimately, primitive goals enable the robot's perceptual and behavioral routines. In total, there are 55 RAPs, five of which are specific to the trash collection task. The RAPs invoke a total of 20 reactive skills, five of which involve visual processing. The visual processing is made specific to the task by the models that are passed to the routines. The RAPs are described in more detail in [Firby *et at.,* 1995].

We'll examine two steps of the method to demonstrate CHIP's visual routines, and cooperative on-board/off-board perceptual and motor processes. For the subgoal *scan-for-object,* CHIP identifies and locates a nearby object that can be segmented from the floor. The subgoal *orient-to-object* consists of precisely lining up with it to grab it.

When searching, CHIP alternates simple exploratory movements (e.g., scanning its head) with calls to a visual routine, the small object finder, which finds and classifies small objects lying on surfaces with only fine (high spatial frequency) texture or none at all. The routine uses the DataCube to grab an image, smooth it, and extract edges. The edges are closed into segments, which are then classified with a set of thresholds on size, aspect ratio, edge density, and gray-scale brightness. The process typically takes under four seconds for a high resolution image of a cluttered scene. Accuracy is currently at 95 percent on a database of challenging test images, and almost perfect in actual runs of the task.

The distance and bearing of the object are computed from a single image using triangulation from the robot's known height and assuming that the object is on the floor. This calibration is cached for particular tilt angles of the camera.

For the goal *orient-to-object,* CHIP aligns itself directly in front of the target, at a distance of less than one

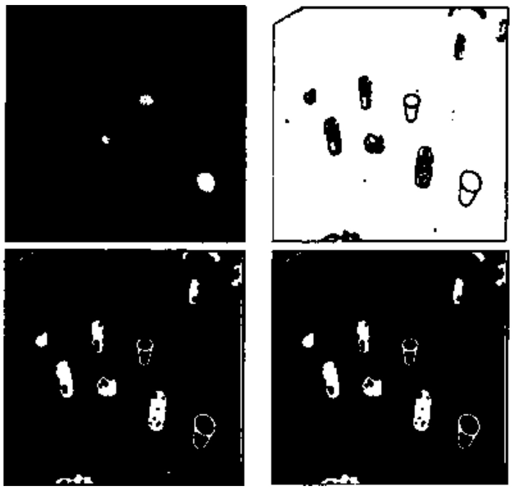
Figure 4: Image processing steps for identifying trash. Upper left: high-resolution test image; upper right: edges from Sobel operator after gaussian smoothing; lower left: edges closed into segments; lower right: small and large segments removed.

meter. A visual tracking routine locates a piece trash repeatedly and generates a target location with which to align. The results are time-averaged to provide a stable and accurate reference point with which CHIP can align.

The tracking routine operates exactly like the object finder described above, except that the identification procedure operates only on a much smaller window (Figure 5), at greater than one update per second. The 2D frame-to-frame image displacements of the tracked object's centroid are fitted to a model of smooth image motion that determines placement of the next processing window [Prokopowicz *et al.*, 1994]. If the object is not in the window, the entire scene is searched. The primary advantage of tracking in this situation is to avoid confused behavior that can result from identifying different items at different stages of the routine.

When the RAP system encouters the goal *orient-to-object*, it selects a method for achieving the goal according to whether the object involved is a piece of trash or a trash can. The method for orienting to a piece of trash expands into two parallel goals: *primitive-track-small-object* and *primitive-orient-to-goal.* The RAP method for *primitive-orient-to-goal* enables a routine to watch for violations of a safety zone in front of the robot (using sonars) and a routine to orient the robot's position according to an input goal location. The method for *primitive-track-small-object* enables the trash-tracking visual routine to update that goal location continuously. The routines enabled for these goals run as separate processes. The orient-to-goal routines run on-board the robot using odometry to drive and turn to a certain distance and direction from the goal location. The trash-tracking visual routine runs off-board and repeatedly computes a relative distance to the target which is turned into a new goal location and passed to the on-board orient-to-goal routine. To orient to the trash can, a different method is chosen by *orient-to-object:* enabling the same action routines used for orienting to trash but a different visual routine for updating the goal location using visual operators better for finding trash cans.

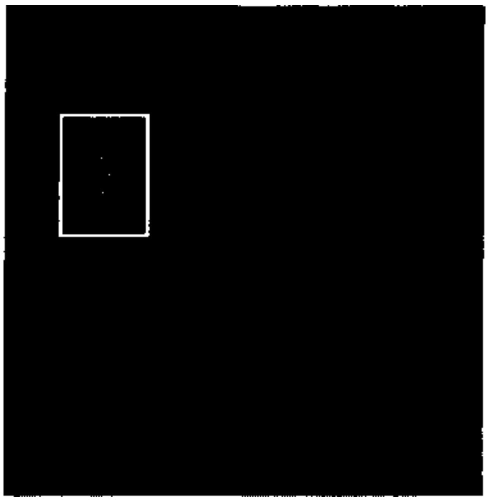
Figure 5: CHIP's view of a can during alignment. Inset: the can as it is tracked. Only the image area in the white box is processed.

Figure 6: left: CHIP rolls forward after aligning with the can; right: CHIP's view as it picks up a can.

Once CHIP has lined up with the object to be picked up, it rolls forward blindly until the LED beam between its fingers is broken by an object, or nothing is encountered within a meter. CHIP is able to align itself and roll forward accurately enough to pick up trash almost every time (Figure 6). Failures are noticed and retried.

The trash finder is designed to find easily-segmented objects known only by their class membership (e.g. pop cans of all types, crumpled paper of various shapes). We have developed a different routine for locating and track-

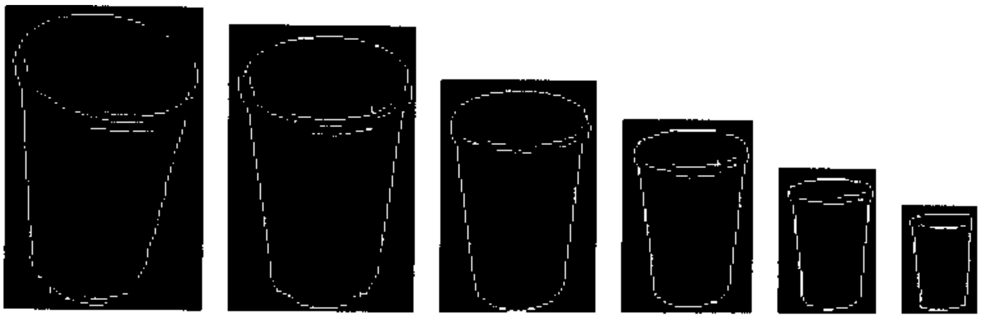

Figure 7: Edge models, taken from distances of 3 meters to 0.75 meters.

ing known objects that cannot necessarily be segmented from their background. We have had some success in this direction using a Hausdorff model-matcher developed at Cornell [Huttenlocher and Rucklidge, 1992].

We assume:

1. color and edge models of the object from likely views (Figure 7)
2. the object sits upright on the floor, and is shorter than the robot
3. the floor is flat
4. the robot's view of the the object suffers at most minor occlusion

This last assumption is less limiting than one might expect, since the robot is free to roam about the room in search of the object.

Using this approach, we are able to avoid searching over all possible views. We search locations in the scene consistent with an object sitting on the floor, and we combine the results of different algorithms to speed the search.

The solution is as follows:

1. Using RAPs, scan the room from left to right, changing view enough to leave a reasonable overlap between views
2. For each view, enable a visual routine that:
   (a) Finds all regions of interest using color [Swain and Ballard, 199l]
   (b) Runs a HausdorfF edge matcher [Huttenlocher and Rucklidge, 1992] in each region of interest

The robot takes advantage of the assumption that the object is on the floor by restricting the region searched vertically, according to the height of the object and the distance to the nearest model in the set. It also takes advantage of this assumption to restrict the range of scale searched by the edge matcher within each region of interest returned by the color matcher. Doing so relies on the distance calibration over the image for the floor at the angle of view of the camera during the search.

The HausdorfF metric allows one to measure the match of a template to an image, allowing minor deformations. The Cornell software searches efficiently over all translations and scalings of the template (within controllable limits) to find the best fit. We have found that the process works somewhat more reliably if the range of scales is limited (to between 100 and 75 percent, in our case) and more models are used to span the scales actually encountered. We have had good results with match criteria that at least 80 percent of the scaled and translated model points lie within one pixel (across or diagonally) of some image point. A reverse match criteria that at least half the image edge points covered by the model lie within one pixel of some model point reduces false matches in dense parts of the edge image.

As efficient as the Cornell software is, we have found it too slow over large, cluttered scenes, except at very low resolutions or with unrealistically tight match criteria. The relatively high complexity of the edge matching motivated our use of color, to screen possible locations prior to edge matching. The histogram back-projection technique assigns to each color pixel a rating according to how well it indicates the presence of the model. The histogram is computed in an 8 bit hue, saturation, value color space, with 3 bits resolution of hue and saturation, and 2 of value. The pixel saliency ratings are then averaged over 4x4 neighborhoods, and the peaks above a threshold are found. An area around the peak is cropped until the smoothed saliency value falls under 10 percent of the peak value. Most of the image has values at or near zero (figure 8L), so the edge matching phase is greatly speeded up. The back-projection and edge-detection is computed on the DataCube, and region cropping is done on the workstation, taking together less than 0.2 seconds. In a typical scene as shown here, matching against all models of a target (we use up to six) takes about 0.5 seconds.

In Table 1 the search times and results are given as the task constraints are added one by one. In all cases, the

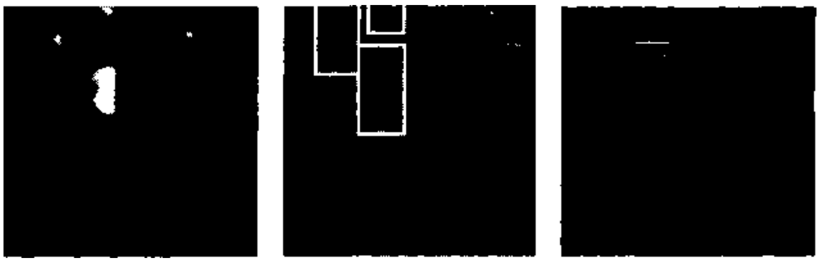

Figure 8: Top left: Back-projected color model of a trash can into medium-resolution image; top right: peaks in image extracted for matching to edge models; bottom: model found.

Constraints	Time	False Positives
Search entire room below horizon	520	4
+ Use color region of interest	25	1
+ Constrain model scale by floor-based distance to region of interest	17	1
+ Verify matched scale against floor-based distance to target	17	0

Table 1: Cost and accuracy of a 360-degree high resolution search for a trash can. Each search used 8 overlapping views. Image grabbing and color processing were done on a Datacube image processor; edge matching on a Sparc 20. Times shown are elapsed seconds, including the time to re-orient the camera.

trash can was correctly identified, but with fewer constraints the search took longer and resulted in more false positives. The technique is reliable, requires only modest computing resources, and can be extended to arbitrary objects - provided the task constraints hold, which are that the object is of restricted size, and sits on the floor in one of a small number of possible orientations.

The distance to an object found with the Hausdorff matcher can be easily computed using the known distance to the original picture of the model and the scale at which the model matched the image. We compare this distance to the triangulated distance judged from the location of the model in the image, assuming that the object is on the floor with the robot. Matches whose distances don't agree within 25 percent are rejected.

The robot has successfully found, picked up, and thrown away hundreds of pieces of trash, at rate of about 12 pieces per hour. We have been able to demonstrate increasing if not perfect reliability. One major impediment to high reliability in our current system is its rudimentary navigation abilities. CHIP does not try to remember where trash has been seen before and ignores small objects on the floor while moving around. These capabilities are being added now.

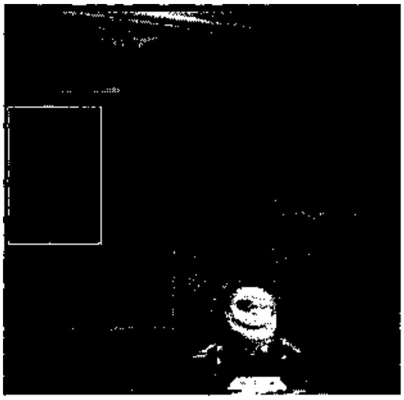

Figure 9: CHIP's view of a trash-can during alignment. Inset: match made to an edge model of the trash-can.

5 Related Work

Kosecka and Bajcsy [Kosecka and Bajcsy, 1993] have studied synthesizing complex behaviors from simpler components. They model their components using Discrete Events Systems (DES) theory, which allows them to predict the *controllability* of composite behaviors, that is, whether or not it is possible to reach a goal state from a given state in the system. We leave the decision of how to combine multiple concurrent skills to the programmer, and instead concentrate on languages to conveniently express which skills should be run in sequence or in parallel, and to build natural interfaces that allow visual operators and control routines to be parametrized and composed to perform a variety of tasks. Aloimonos [Aloimonos, 1990; 1994] has been an outspoken advocate of what he calls the *purposive* approach to achieving general-purpose vision, which is generally consistent with the approach we have taken here.

6 Conclusion and Future Work

This paper describes the design of an implemented architecture that both enables the use of context from the task and environment to simplify vision problems, and allows the components (visual operators and control routines) to be reconfigured to solve a variety of tasks with minimal reprogramming. It provides a framework for building vision systems that are robust and timely, *and* can be utilized for a variety of tasks. We have demonstrated an implemented system (important parts of which are available to others) that works on a trash collection task in an office environment, and have explained why the components can be re-used in another task or environment.

We are using the architecture to encode a number of different robot skills, including fetching tools and other objects under human guidance, and delivering items to people's desks. But one of the first tasks is to prepare for the next robot competition, in which the task will be to expanded to sorting recyclables from the trash. Reprogramming CHIP to perform this task will involve merely acquiring models for the recycling bin, and modifying the put-trash-in-can RA P to choose a destination depending on the identity of the object picked up.

References

[Agre and Chapman, 1987] Philip E. Agre and David Chapman. Pengi: An implementation of a theory of activity. In *Sixth National Conference on Artificial Intelligence,* Seattle, WA, July 1987. AAAI.

[Aloimonos, 1990] John Aloimonos. Purposive and qualitative active vision. In *International Conference on Pattern Recognition,* pages 346-360, 1990.

[Aloimonos, 1994] Yiannis Aloimonos. What I have learned. *CVGIP: Image Understanding,* 60:74-85, 1994.

[Bajcsy, 1988] R. Bajcsy. Active perception. *Proceedings of the JEEE,* 76:996-1005, 1988.

[Brooks, 1986] Rodney A. Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation,* RA-2:14-23, 1986.

[Brown, 1994] Christopher M. Brown. Toward general vision. *CVGIP: Image Understanding,* 60:89-91, 1994.

[Dickmanns and Graefe, 1988] Ernst Dieter Dickmanns and Volker Graefe. Dynamic monocular machine vision. *Machine Vision and Applications,* 1:223-240, 1988.

[Firby *et ai,* 1995] R. James Firby, Peter N. Prokopowicz, and Michael J. Swain. Plan representations for picking up trash. Technical report, Department of Computer Science, University of Chicago, 1995.

[Firby, 1987] R. James Firby. An investigation into reactive planning in complex domains. In *Sixth National Conference on Artificial Intelligence,* Seattle, WA, July 1987. AAAI.

[Firby, 1989a] R. James Firby. Adaptive execution in complex dynamic worlds. Technical Report YALEU/CSD/RR 672, Department of Computer Science, Yale University, 1989.

[Firby, 1989b] R. James Firby. Adaptive execution in complex dynamic worlds. Technical Report YALEU/CSD/RR #672, Computer Science Department, Yale University, January 1989.

[Firby, 1994] R. James Firby. Task networks for controlling continuous processes. In *Proceedings of the Second International Conference on AI Planning Systems,* pages 49-54, 1994.

[Horswill, 1993] Ian D. Horswill. *Specialization of Perceptual Processes.* PhD thesis, Dept. of EE & CS, Massachusetts Institute of Technology, 1993.

[Huttenlocher and Rucklidge, 1992] Daniel P. Huttenlocher and William J. Rucklidge. A multi-resolution technique for comparing images using the hausdorff distance. Technical Report CUCS TR 92-1321, Department of Computer Science, Cornell University, 1992.

[Kahn *et ai,* 1994] Roger E. Kahn, R. James Firby, and Michael J. Swain. The message hub. Animate Agent Project Working Note 4, University of Chicago, April 1994.

[Kahn, 1993] Ari Kahn. Using the datacube client class. The University of Chicago, Computer Science Department, 1993.

[Kosecka and Bajcsy, 1993] Jana Kosecka and Ruzena Bajcsy. Cooperation of visually guided behaviors. In *IEEE International Conference on Computer Vision,* pages 502-506, 1993.

[Marr, 1982] D.C. Marr. *Vision.* Freeman, 1982.

[Nelson, 1989] R. C. Nelson. Using flow field divergence for obstacle avoidance in visual navigation. *IEEE Transactions on Pattern Analysis and Machine Intelligence,* 11:1102-1106, 1989.

[Prokopowicz *et al,* 1994] Peter N. Prokopowicz, Michael J. Swain, and Roger E. Kahn. Task and environment-sensitive tracking. In *Proceedings of the I APR/IEEE Workshop on Visual Behaviors,* 1994.

[Slack, 1990] Marc G. Slack. Situationally driven local navigation for mobile robots. Technical Report JPL Publication 90-17, Jet Propulsion Laboratory, April 1990.

[Slack, 1992] M.G. Slack. Sequencing formally defined reactions for robotic activity: Integrating raps and gapps. In *Vol. 1828 Sensor Fusion V: Simple Sensing for Complex Action,* Boston, MA, November 1992. SPIE.

[Swain and Ballard, 199l] Michael J. Swain and Dana H. Ballard. Color indexing. *International Journal of Computer Vision,* 7:11-32, 1991.

[Tarr and Black, 1994] Michael J. Tarr and Michael J. Black. A computational and evolutionary perspective on the role of representation in vision. *CVGIP: Image Understanding,* 60:65-73, 1994.

[Ullman, 1984] Shimon Ullman. Visual routines. *Cognition,* 18:97-159, 1984.