

Programming CHIP for the IJCAI-95 Robot Competition

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

■ The University of Chicago's robot, CHIP, is part of the Animate Agent Project, aimed at understanding the software architecture and knowledge representations needed to build a general-purpose robotic assistant. CHIP's strategy for the Office Cleanup event of the 1995 Robot Competition and Exhibition was to scan an entire area systematically and, as collectible objects were identified, pick them up and deposit them in the nearest appropriate receptacle. This article describes CHIP and its various systems and the ways in which these elements combined to produce an effective entry to the robot competition.

The University of Chicago's robot, CHIP, is part of the Animate Agent Project, aimed at understanding the software architecture and knowledge representations needed to build a general-purpose robotic assistant (Firby et al. 1995). The Animate Agent Project defines two primary levels of knowledge representation: (1) the skill level consisting of modular processes that can be configured in a variety of ways and (2) the reactive task-execution level consisting of a goal interpreter and a library of reactive action packages (RAPs), or *reactive plans*, that describe ways of achieving system goals.

One of the primary goals of the Animate Agent Project is to develop a vocabulary of skills and plans that can be used to achieve a wide variety of everyday tasks. Thus, while addressing the Office Cleanup event of the 1995 Robot Competition and Exhibition, we were attempting to build generic skills and RAPs that could be reused as building blocks in RAPs for future tasks.

CHIP went into the cleanup competition with a general-purpose strategy for cleaning up a room. Broadly, the strategy was to scan

an entire area systematically and, as collectible objects were identified, pick them up and deposit them in the nearest appropriate receptacle. This plan for cleaning up a room is built from a collection of more general-purpose operations, such as find-object-type, move-to-target, and pick-up-target. In this sense, only CHIP's highest-level goals were programmed specifically for the robot competition, which was held in conjunction with the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI-95); the rest are generic and are the building blocks for CHIP's future behavior.

CHIP the Robot

CHIP is the working robot at the Animate Agent Laboratory at the University of Chicago. As illustrated in figure 1, CHIP is built on a Real-World Interface three-wheeled synchro-drive mobile base. This base is surrounded by an octagonal bumper and supports a body roughly three feet high. Around the middle of the body is a ring of eight sonar sensors and on top is a pair of color cameras mounted on a computer-controlled pan-tilt platform. Mounted in front of CHIP's body where it can reach the floor is a Heathkit Hero robot arm that has been augmented with force and contact sensors to get tactile feedback from the gripper. On board, CHIP carries a 68000 computer to manage the sensors and control the arm and a 68030 computer to run the action skills discussed later. Video from CHIP's cameras is broadcast off board to a DataCube image-processing system attached to a SPARC-20 workstation, where all vision processing is done.

The software used for CHIP consists of a

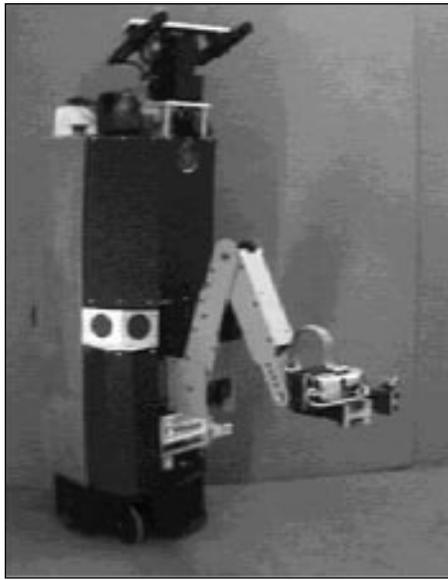


Figure 1. *CHIP the Robot.*

variety of connected systems. The low-level motor and sensor-processing software is written in C and runs on board. The concurrent, perceptual-motor skills that make up CHIP's modular control system are written in CRL (Firby 1994a), with some running on board the robot and others running off board on a MACINTOSH computer. The on-board and off-board components are linked by a radio Ethernet. The RAP system (Firby 1995, 1994b, 1989) for task-level sequencing of the modular skills is written in LISP and runs off board on the MACINTOSH where it interfaces with the CRL skill system. Vision processing (Kahn and Swain 1995) is done off board on the SPARC-20 and communicates with other skills and the RAP system on the Ethernet. The vision system, the perceptual-motor skills, and the RAP task-execution system make up the animate agent architecture for intelligent, real-time robot control, as shown in figure 2.

The Animate Agent Architecture

The RAP system is designed to deal with achieving goals in a dynamic environment. Each RAP task description encodes a set of methods for carrying out the task in different situations, a success check to tell when the task has accomplished its purpose, and notations that describe when things are not going as expected. At run time, a RAP task examines its methods and selects one that is appropriate in the current situation. By doing method selection at run time, RAPs are more likely to select the best method, even if the world is changing or contains details that cannot be predicted in advance. RAP methods can also include other RAP tasks as subgoals, and the resulting hierarchy helps the system to cope with complexity by specializing actions through many different levels of detail. RAP methods also include annotations that specify assumed constraints and expected subtask results. These annotations are monitored during the execution of the method and cause the method to fail when they are not met. Monitoring expectations helps the system to recognize early that a selected method has become invalid because of changing circumstances or unexpected events. To further increase robustness, a RAP task includes a success test to confirm that a method has succeeded in achieving its goal. If the success test is false after a method has completed, the RAP task continues to run, selecting alternative methods until the success test is satisfied, or there are no methods to try. This tenacious execution allows a task to recover when the

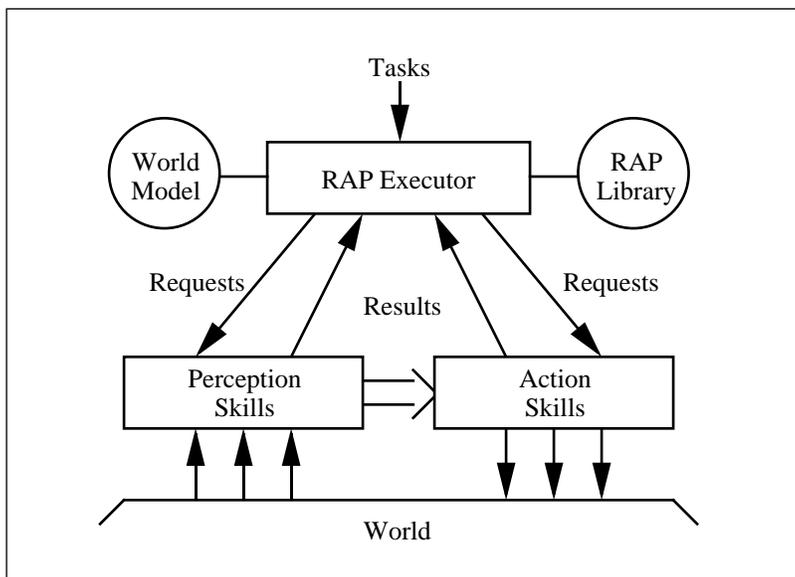


Figure 2. *The Animate Agent Architecture.*

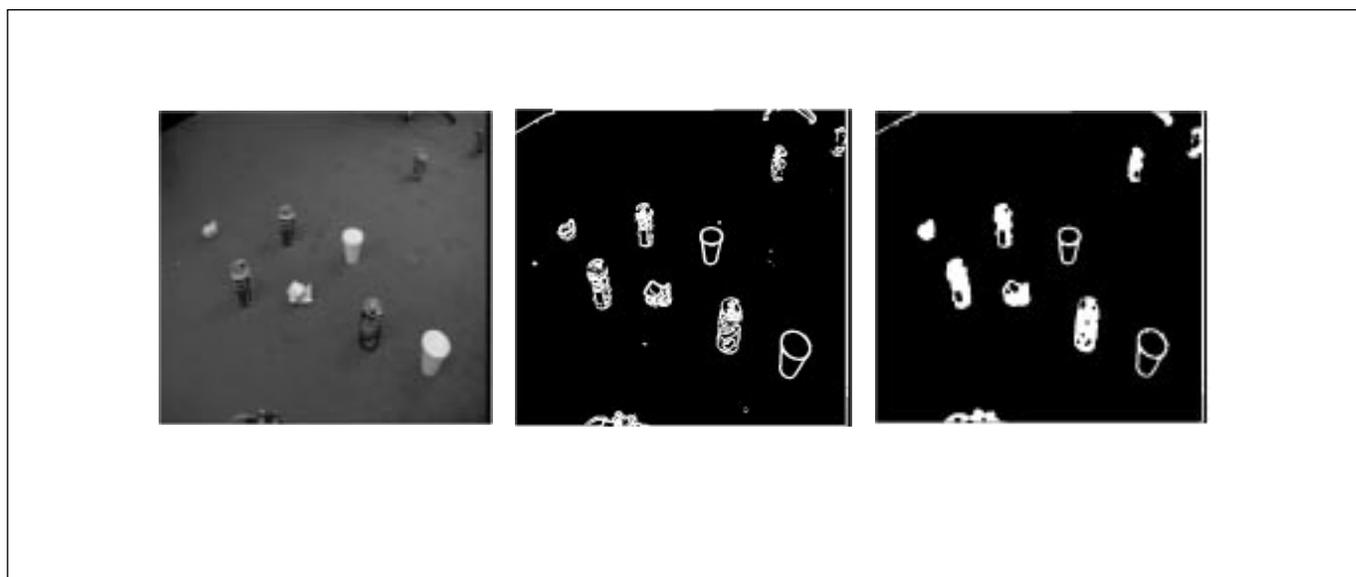


Figure 3. Trash: Raw Image, Edge Image, and Filtered Image.

information it uses to select a method is incorrect, when actions don't have their intended effects, or when the situation changes while a method is executing.

The skill system is designed to deal with soft real-time control of the robot's actuators. The skill system is a network of sensing and action processes, defined by separate programs, that can be enabled or disabled at will. When a skill is enabled, it receives a set of parameters and begins to run. Running skills can access current sensor values, control actuator set points, and communicate with one another through global channels. The actual behavior of the robot at any given moment is the result of the enabled skills acting on input from the environment. Thus, the behavior of the robot (that is, its apparent immediate goal) can be changed by changing the set of active processes.

The RAP system refines tasks into commands that program the skill system. These commands enable skills and connect them into control networks to carry out activities in the situations encountered at run time. Goal-directed behavior results from task refinements that generate a sequence of different skill-system configurations.

Basic Skills

The basic skills used by CHIP during the competition can be broken down into *perception skills* for extracting visual information from the world and *action skills* for effecting changes in the world. While CHIP is carrying

out a given activity, a combination of these skills will typically be running concurrently and working together to control the robot.

Perception

One of the key features of the architecture that makes the trash cleanup task possible is the ability to easily use different sensing strategies for different tasks. This feature is particularly important when building a system that uses vision for a wide variety of tasks because different tasks will typically require visual data to be processed in different ways. For the office cleanup task, CHIP used three different vision-processing algorithms: (1) an algorithm for finding and classifying small objects on the floor, (2) an algorithm for tracking a selected small object, and (3) an algorithm for finding objects against more complex backgrounds.

Finding and Tracking Trash CHIP can recognize a broad class of small objects when seen against a relatively clean background. The algorithm for this process has four steps: First, an edge operator is run over the visual scene to create an edge image, which is then segmented into regions that might be possible objects (figure 3). Second, the size, aspect ratio, edge density, average color, fraction of white, and contour regularity are computed for each possible object. Third, the resulting feature vector for each object is classified against a set of fuzzy exemplars by choosing the nearest neighbor within a maximal distance. We entered the competition with



Figure 4. Tracking Trash: Target and Next Search Area.

exemplars for generic cans that could handle essentially any drink can, cups (either white or colored), and balls of paper—a vestige of the 1994 competition. Finally, the relative location of each object with respect to the robot is calculated using the camera height, orientation with respect to CHIP's body, and the location of the object in the image. The end result is that CHIP knows the type and the location of all the pieces of trash in the current scene.

However, the location calculated for each piece of trash is only approximate. Calibration errors and low visual resolution combine to make a trash location more uncertain the farther away it is. Furthermore, as CHIP moves toward a piece of trash, actuator error has the effect of increasing the uncertainty of the piece's location. To compensate for these errors, CHIP must track the piece of trash it is approaching while it is moving.

The visual process for tracking a piece of trash is simpler than for identifying trash because we want the process to run as fast as possible. The trash tracker uses two heuristics to reduce processing time. First, CHIP knows roughly where the piece of trash is supposed to be (from the previous algorithm), so the tracking algorithm only looks at a small portion of the image around that area (figure 4). Second, the trash tracker only performs the first step of the trash-finding algorithm, computing the edge image over the small region

and segmenting out a single object region. The region is assumed to correspond to the piece of trash being tracked, and the tracker returns its location. This process is repeated as fast as possible so that as CHIP moves, the piece of trash stays within the small search region of the tracker. If the tracker fails to find a possible segmented object in the small search region, it doubles the size of the search region and tries again, repeating the process until it finds an object, or it has searched the whole scene.

Finding Trash Cans During the competition, trash cans could not be segmented easily from the background using edge data alone because they were along the walls in the office near corners, furniture, and other objects. To find and recognize objects that are not easily separated from the background, CHIP used a two-step algorithm, illustrated in figure 5. First, the scene was examined using color-histogram back projection (Swain and Ballard 1991) to find areas containing the same colors as the object. An edge operator was then applied to these areas followed by a soft edge-template-matching algorithm that includes adjustments for viewpoint transform. This algorithm uses the Hausdorff distance (Huttenlocher and Rucklidge 1992) to compare different possible template matches, and the best match gives the exact location, size, and pose of the object. For the competition, we used the distinctive colors of the trash can and recycling bin labels to locate possible cans and a set of six known views of the object to match against (figure 6). Unfortunately, even using fast color back projection to reduce the area searched by the Hausdorff template matcher, this algorithm took several tens of seconds.

The same problems of low resolution and calibration error that arise when finding trash also arise when finding trash cans. Trash can locations from the template-matching routine are only approximate and must be refined as CHIP gets closer and lines up to drop in a piece of trash. For this process, the same template-matching skill is used but at a much lower resolution. Starting with a lower-resolution image speeds the matching process significantly but prevents CHIP from seeing the trash can unless it is close at hand.

Action

The basic skills that CHIP uses for taking action fall into three categories: (1) skills for moving CHIP's arm, (2) skills for moving CHIP from one place to another, and (3) skills for moving the pan-tilt head that carries CHIP's cameras.

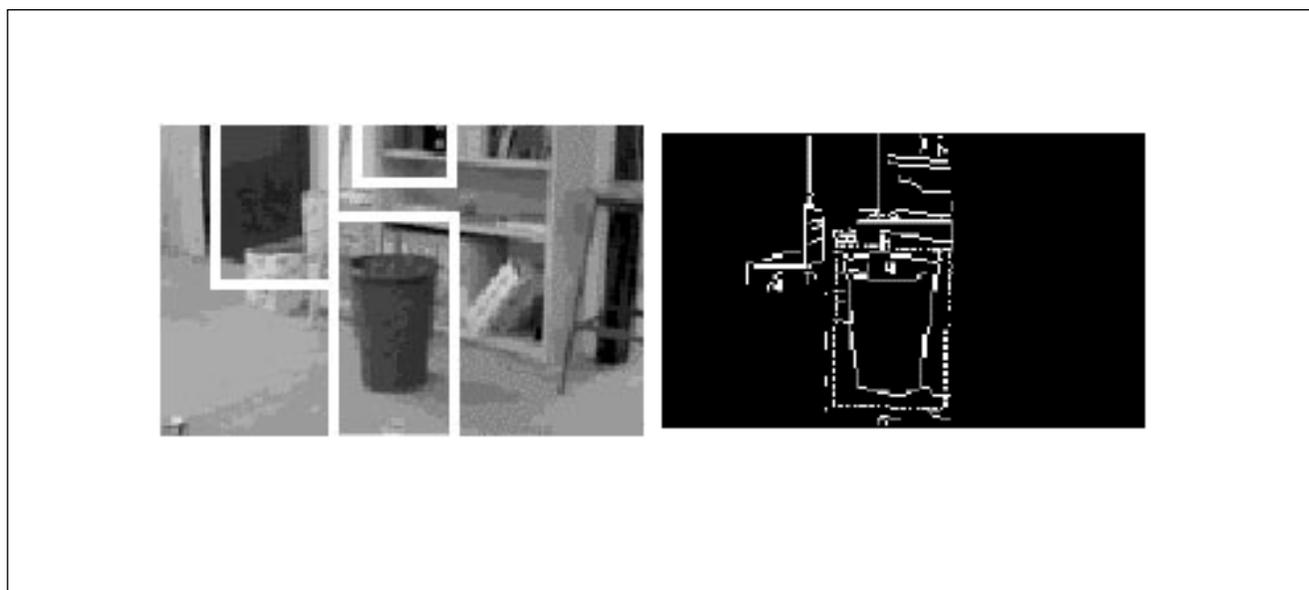


Figure 5. Trash Cans: Original Scene with Color Areas Boxed, Edges for Areas of Interest, and Hausdorff Match.

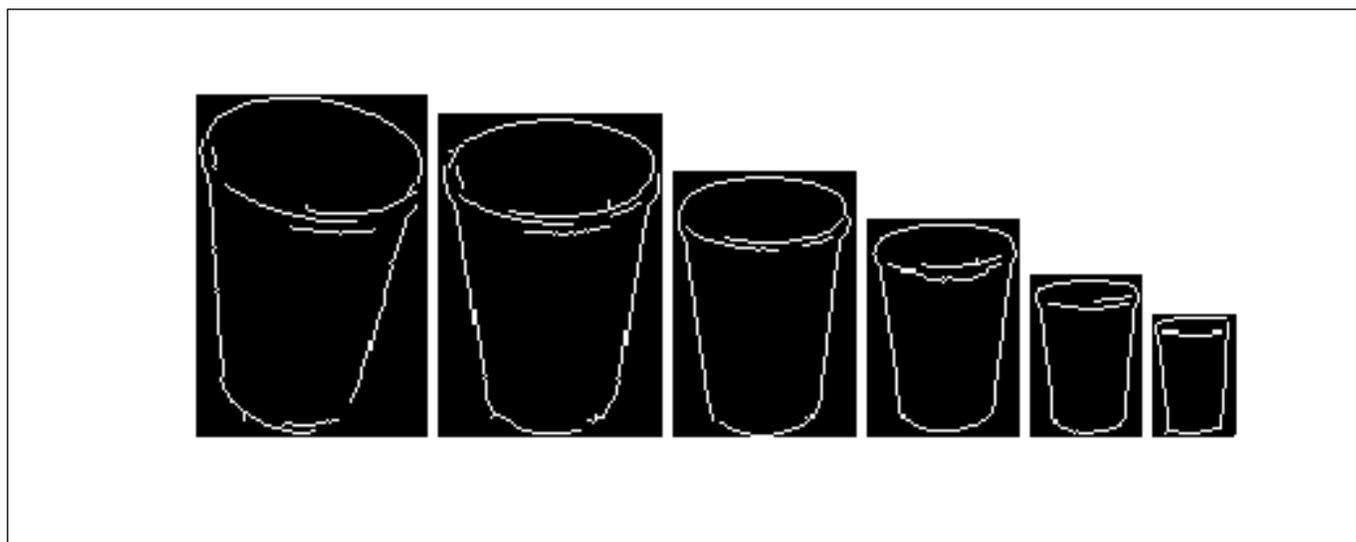


Figure 6. Trash Can Hausdorff Models.

Manipulating Objects The skills used to control CHIP's arm include a skill to move the wrist to a given point in front of CHIP, a skill to tilt and twist the wrist, and several skills to open and close the gripper. Multiple gripper skills are used because of the variety of different stopping conditions required in different situations. For example, a generic skill for setting the gripper width to a specific value is used to open the gripper in preparation for picking up a piece of trash. This skill will open or close the gripper to get to the desired width and will signal when the goal is reached or when the gripper gets stuck and

can't move. Another skill is used to grasp objects by closing the gripper until a specific pressure is generated. A different skill is used to let go of an object by opening the gripper until either its finger sensors signal the gripper is empty, the gripper is open as far as it will go, or the gripper gets stuck and can't move. A fourth skill is used to close the gripper until the fingers touch so that the arm can be tucked away while CHIP is moving with its hand empty. The skills associated with CHIP's arm do not currently use visual feedback or account for obstacles in the arm's work space.

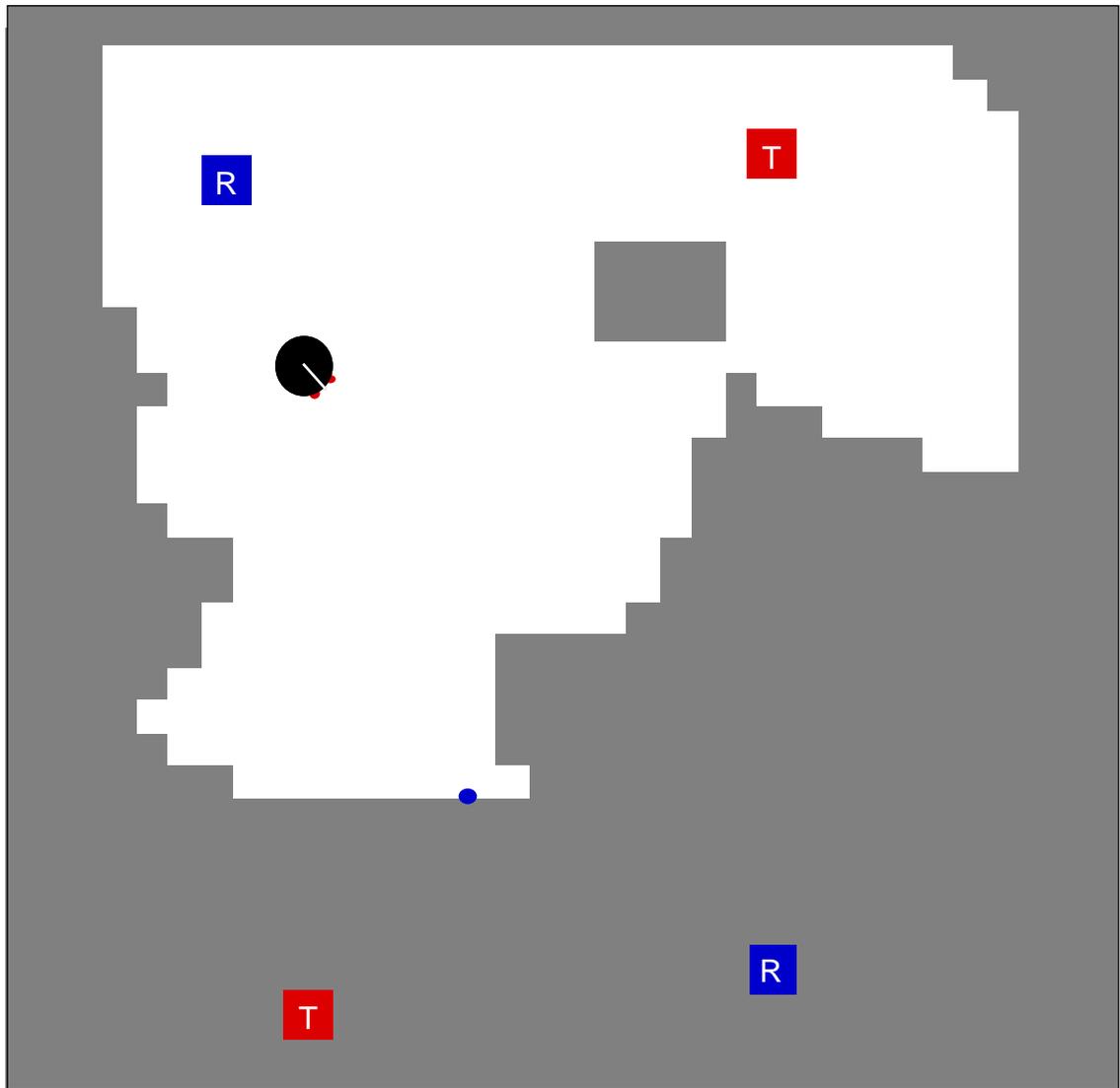


Figure 7. Areas Searched for Trash during Competition.

The trash cans and recycling bins are marked as squares. A known can is also shown as a dot.

CHIP's arm can only move in a vertical plane directed forward; so, CHIP must be lined up precisely with objects on the floor to be able to reach them. This requirement is met by a special skill for aligning the center of CHIP's body with a target location at a precise standoff distance. The skill signals when the alignment is good or when the robot cannot move without hitting something. This skill relies on other skills to generate the target location, and it can be used to align with a variety of objects or even a dead-reckoned location. We use the trash-tracking skill to align with a piece of trash and the low-resolution Hausdorff template-matching skill to align with a trash can.

Once CHIP is aligned with a piece of trash, the arm is down near the ground, and the gripper is open, another skill is used to move CHIP straight forward until the gripper detects an object between its fingers. This skill signals completion when an object is detected in the gripper, when a certain distance has been traversed without detecting an object, or when the fingers bump into something.

Moving to Locations CHIP uses two basic skills for moving from place to place: (1) one to turn to a particular heading and (2) one to move to a particular location. The turn skill turns CHIP in the appropriate direction and signals completion when CHIP is facing the right way. It also monitors the sonar sensors

and moves CHIP backward, if possible, to avoid bumping the arm while turning. If CHIP cannot move without bumping the arm or backing into something, the skill signals that CHIP is stuck. The move skill drives CHIP toward the goal location, monitors sonar values, and uses a simple potential field obstacle-avoidance strategy as it goes. This skill signals when CHIP reaches the goal or when the obstacle-avoidance strategy gets stuck. Like the aligning skill, both turning and moving rely on other skills to supply the target for their actions. In this manner, different target-generating skills can be used to get CHIP to perform different actions at different times. For example, a target generated by dead reckoning can be used to move CHIP to a particular location in space, or a visual-tracking skill can be used to get CHIP to approach a specific object and follow it if it moves.

Moving the Head Skills for moving the pan-tilt head come in both feedback and nonfeedback-driven varieties. There is a skill for moving the head to a specified pan, which signals when the movement is complete, and a skill for panning the head to a goal direction generated by another skill. Combining this latter skill with a visual-tracking skill keeps the head pointed toward a moving target. There are similar skills for controlling tilt.

Map Building and Object Memory

The basic algorithm used by CHIP for picking up trash is to look for a piece of trash, pick it up, look for a trash can, drop off the trash, and repeat. However, we also want CHIP to remember the locations of trash and trash cans and be able to tell when all the trash is gone. In a small office such as that used in the competition, people don't need to remember where trash is because they can quickly scan the entire floor to see if any is around. The same scan also tells them when the task is done because the entire floor is clean. However, CHIP is nearsighted and cannot accurately identify a piece of trash more than eight feet away. Therefore, CHIP must rely on memory to know where trash has already been seen and which areas of the office have already been cleaned up.

Mapping Areas Searched

To remember where CHIP has looked for trash and trash cans, a binary grid in global coordinates is maintained for both trash and cans. Each time CHIP uses the visual skill for finding

trash, the wedge of floor space imaged by the skill is recorded in the grid by setting the points covered by the image. The grid is then used by CHIP to search the office systematically for trash. When CHIP knows of no piece of trash on the floor, it consults the grid to find the nearest area that has not yet been examined. CHIP then moves to bring this area into view and execute the trash-finding visual skill. The visible area is updated in the grid, and the next time CHIP is searching for trash, a different, unexplored area is selected to look at. Simple heuristics are used to prefer to explore areas that can be seen by turning the pan-tilt head; then those that can be explored by turning the robot; and, finally, those that require moving to a new vantage point. A similar algorithm is used for keeping track of where trash cans have been sought. The grid showing where CHIP looked for trash during the final event of the competition is shown in figure 7.

Mapping Accessible Areas

In addition to keeping track of where visual skills have looked, CHIP uses a simple occupancy-grid technique to map office obstacles using sonar data (Borenstein and Koren 1991; Elfes 1989). This mapping happens passively, in parallel with the basic task of cleaning the office, and results in the construction of a map of reachable areas. This map is used to ensure that all vantage points chosen by the algorithm for looking for trash and trash cans are actually accessible. Furthermore, because the office environment was sealed by walls and closed doorways, once CHIP had checked every accessible area for trash, it knew that no more trash existed in the office, and the task was complete. During the competition, CHIP further constrained where it needed to look for trash using the knowledge that the office held four trash cans in the four corners. Thus, once three trash cans were found, a rectangle enclosing those cans marked the area enclosed by the office. This heuristic also helped to tell CHIP when to stop cleaning up, even if a door was open, and there was accessible space outside the office area. The occupancy grid generated during the final event of the competition is shown in figure 8.

Remembering Item Locations

Of course, along with remembering those areas that have been explored, CHIP also remembers where it has seen trash and trash cans. The locations of these items are recorded in a simple database that supports looking up the closest item of a particular type. CHIP

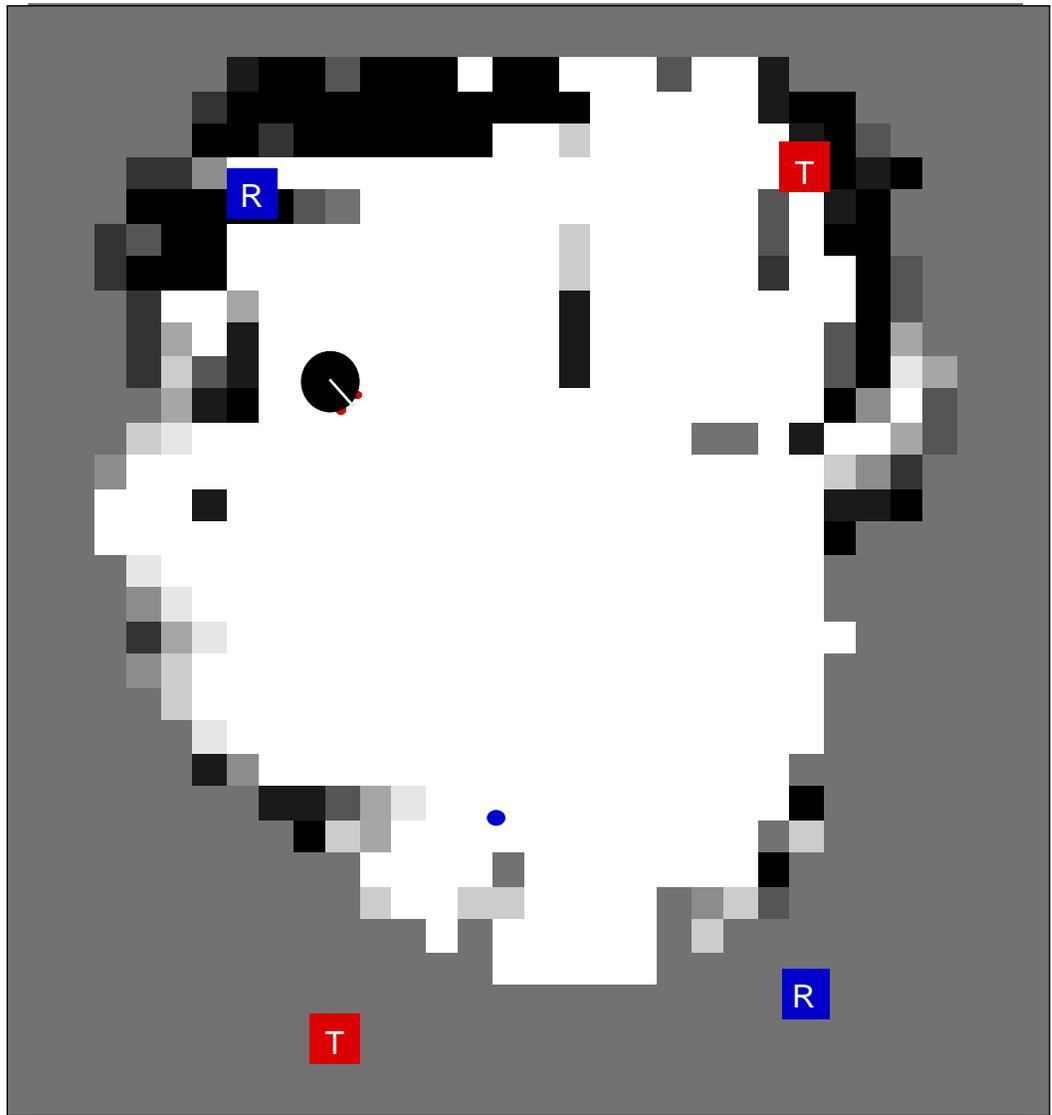


Figure 8. Occupancy Grid Generated during Competition.

then uses the heuristic of picking up trash it knows about before exploring new areas.

Unlike pieces of trash, CHIP can see trash cans all the way across the office and knew that there were exactly two trash cans and two recycling bins. This information led to the heuristic that CHIP use a known trash can if it was facing toward one after picking up a piece of trash or that it look around first if it was facing away from a known can, and some trash cans had not yet been found. The result of this heuristic was that CHIP found all four trash cans early and then always moved to the closest can of the appropriate type. The

locations of trash pieces and trash cans were recorded in the database as a direct result of using the corresponding visual routines.

Putting It All Together

The skills and mapping processes described earlier are not enough to do the trash cleanup task by themselves. The RAP task-execution system is used to decide what action CHIP should perform next and, hence, chooses the sequence of active skills that actually control CHIP's activities.

The RAP library used for the cleanup task is

organized as much as possible around reusable subtasks that can be used in future tasks as well. These subtasks include moving the robot from place to place, searching for items in the world, manipulating objects, and doing the trash cleanup task itself.

Moving to a New Location

The basic RAP plans for moving CHIP to a given goal location consist of sequences of the turning, moving, and tracking skills described previously. These plans are variations on the following sequence:

Step 1: Enable a skill to track the goal location using odometry and dead reckoning. Use this skill as the target generator for the skills to follow.

Step 2: If CHIP is not facing within 30 degrees of the goal location, enable the turn skill to turn CHIP in roughly the correct direction. Signal completion when facing the correct way.

Step 3: Enable the skill to move CHIP toward the goal location. Signal completion when CHIP is close to the goal.

Step 4: If CHIP is not facing toward the goal, enable the turn skill again to turn CHIP in the correct direction. Signal completion when facing the correct way.

Step 5: Disable the dead-reckoning skill that generates the target location for the motion skills.

For the office cleanup task, all goal locations are either object locations from the item database or vantage points for areas yet to explore. However, in other situations, different goal-tracking skills can be used in step 1 to get CHIP to follow a visual target or approach a moving object.

These RAP methods can fail if the goal location cannot be reached. In particular, when moving to an unexplored area, CHIP might encounter new obstacles that prevent achieving the goal. The movement skill recognizes this condition by monitoring progress toward a goal and signaling when progress is not being made. The RAPs used in the competition simply fail in this case.

Searching for Trash and Trash Cans

One of the key aspects of the office cleanup task is the need to search the world for trash and trash cans. This behavior is encoded in CHIP as a collection of RAPs that use the spatial memory described previously to execute the following basic plan:

Step 1: Select an unexplored area nearby using the memory of areas already explored looking for this object type.

Step 2: Use the movement subtasks described earlier to move to a good vantage point for viewing the area.

Step 3: Execute the appropriate visual skill for finding the desired item, and continue when the skill completes.

Step 4: If an item of the correct type was added to memory, succeed; else, repeat.

Steps 1 and 4 in this plan are actions on memory, step 2 is an instantiation of a RAP subtask, and step 3 enables a visual skill and waits for its completion.

This basic method can fail if there are no unexplored, accessible areas for the object or if the selected area cannot be reached. In the first case, the searching RAP simply reports failure because there is nowhere left to look. In the second case, the searching RAP catches the movement failure and simply repeats. The mapping processes will have detected the obstacles blocking progress, and the next iteration of the RAP will select a new area that is accessible based on this new information as well.

Picking Up and Dropping Off Trash

Once a piece of trash or a trash can has been found and approached, the next step is to pick up the trash or drop it off. The sequence of activities required to pick up a nearby piece of trash are as follows:

Step 1: Begin tracking the piece of trash with the visual trash tracker, and use output of this skill as the goal for aligning the robot near enough for a good grasp.

Step 2: Lower the arm near the ground, keeping the wrist horizontal, and open the gripper as wide as possible.

Step 3: Move forward until the object comes between the fingers of the gripper.

Step 4: Close the gripper until an appropriate pressure is reached.

Step 5: Move the arm up into the position for holding the trash while moving.

The first step is accomplished with a collection of concurrent skills for tracking the object and orienting the robot. The next step is really three steps done in parallel by activating concurrent skills for moving the arm, the wrist, and the gripper. The third, fourth, and fifth steps each correspond to activating individual skills and waiting for them to signal completion or some problem. As with the movement RAPs, this plan is parameterized by object type, and step 1 can select different tracking skills for different object types. Therefore, the plan can be used for a variety of objects besides trash.

Possible problems with this basic plan are

By treating all these failures as the same, CHIP will “forget” about problem pieces of trash and never try to pick them up again.

that the trash tracker might lose the target object, moving forward might miss placing the object in the gripper, or closing the gripper might squeeze the object out from between the fingers. In the first case, losing track of the object probably means that someone snatched it away; so, the object is removed from the memory database, and the RAP fails. The second case can also be the result of someone taking the object, or it can be an alignment error. If someone took the object, it should be removed from memory, but if it's an alignment error, CHIP should try to pick it up again. At the competition, we treated both of these cases as if the object had vanished and removed it from the database. We also did the same thing in the third case, when the gripper closes without actually getting the object. By treating all these failures as the same, CHIP will “forget” about problem pieces of trash and never try to pick them up again. What should happen is that the area of the world in which a grasp failure occurs should be reexamined after the failure to see if the piece of trash in question is still there.

A similar sequence of RAP steps is used to drop the trash into the trash can once the can is nearby:

Step 1: Begin tracking the trash can with the low-resolution template tracker, and use output of this skill as the goal for aligning the robot.

Step 2: Raise the arm high enough to clear the trash can, keeping the wrist horizontal.

Step 3: Move forward enough to remove the offset between the hand and the trash can put there by the aligning skill.

Step 4: Open the gripper until nothing is between the fingers.

Step 5: Back up the same distance moved forward.

Step 6: Move the arm and wrist down, and close the gripper to get things ready for looking around again.

Again, each of these steps corresponds to a set of one or more skills being enabled to actually control the activity.

This plan is subject to the same potential tracking failure as the pickup plan while it is aligning with the trash can. If the tracker loses its target, the trash can in question is assumed to have moved or, more likely, to have been something incorrectly recognized as a trash can. Thus, the trash can is removed from the memory database, and the method fails. Another possible error is that the piece of trash being held does not drop from the gripper when it is opened. With a piece of trash stuck in its gripper, CHIP is really in trou-

ble and simply asks for help.

The Complete Cleanup Task

The top-level RAP methods for the cleanup task define the following plan:

Step 1: Select a piece of trash. If one is known in the database, use it; else, search.

Step 2: Go to the trash.

Step 3: Pick it up.

Step 4: Select an appropriate trash can. If a good one is known, use it; else, search.

Step 5: Go to the trash can.

Step 6: Drop off the trash.

Step 7: Repeat.

All these steps are parameterized by object type and correspond to the instantiation of RAP tasks already described. In particular, the tasks of going to a piece of trash, a trash can, or a search vantage point all instantiate the same RAP methods. Selecting and searching for trash and trash cans also use the same underlying methods.

This basic plan can have a number of problems: No pieces of trash might be found, a piece of trash might be unreachable, pickup might fail, the trash might be dropped while CHIP moves, or there might not be an accessible trash can. The first problem is caught by this plan as a failure of the first step, and it means the cleanup task is complete. The second problem is dealt with by removing the piece of trash from the memory database and going back to step 1. If pickup fails, or the piece of trash is dropped, the plan simply restarts at step 1. If no trash cans can be found, then the plan fails and so does the cleanup task itself.

Summary

This task required implementing algorithms for robot navigation; simple object manipulation; and visual sensing for finding, identifying, and tracking objects on the floor. It also required integrating these pieces into coherent plans to systematically pick up all the trash in the office and deal with contingencies such as missing a grasp, dropping something that is being carried, or seeing ghost trash or trash cans because of sensing errors. The resulting implementation used 34 skills and 80 RAPs. Of these, only three top-level RAPs were specific to the cleanup task. We believe most of the RAPs and skills are reusable in other tasks because they encode relatively generic actions such as finding an object, picking an object up, and moving to a new location while coping with changes and failures in a real-world environment.

References

Borenstein, J., and Koren, Y. 1991. The Vector Field Histogram—Fast Obstacle Avoidance for Mobile Robots. *IEEE Transactions on Robotics and Automation* 7(3): 278–288.

Elfes, A. 1989. Using Occupancy Grids for Mobile Robot Perception and Navigation. *IEEE Computer* 22(6): 46–57.

Firby, R. J. 1995. The RAP Language Manual, Animate Agent Project Working Note, AAP-6, Version 1, Computer Science Department, University of Chicago.

Firby, R. J. 1994a. The CRL Manual, Animate Agent Project Working Note, AAP-3, Version 1, Computer Science Department, University of Chicago.

Firby, R. J. 1994b. Task Networks for Controlling Continuous Processes. In *Proceedings of the Second International Conference on AI Planning Systems*, 49–54. Menlo Park, Calif.: AAAI Press.

Firby, R. J. 1989. Adaptive Execution in Complex Dynamic Worlds, Technical Report, YALEU/CSD/RR #672, Computer Science Department, Yale University.

Firby, R. J.; Kahn, R. E.; Prokopowicz, P. N.; and Swain, M. J. 1995. An Architecture for Vision and Action. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, 72–79. San Francisco, Calif.: Morgan Kaufmann.

Huttenlocher, D. P., and Rucklidge, W. J. 1992. A Multi-Resolution Technique for Comparing Images Using the Hausdorff Distance, Technical Report, CUCS TR 92-1321, Department of Computer Science, Cornell University.

Kahn, R. E., and Swain, M. J. 1995. Understanding People Pointing: The PERSEUS System. In *Proceedings of the International Symposium on Computer Vision*, 569–574. Washington, D.C.: IEEE Computer Society.

Swain, M. J., and Ballard, D. H. 1991. Color Indexing. *International Journal of Computer Vision* 7:11–32.



R. James Firby is an assistant professor in the Department of Computer Science at the University of Chicago. He received his Ph.D. in computer science from Yale University in early 1989 and worked as a member of the technical staff at the Jet Propulsion Laboratory through 1990. Firby's primary research interest is the development of intelligent agents designed to work with humans in complex environments. His work in this area is embodied in several projects aimed at understanding activities that require the real-time integration of sensing, reactivity, goal-directed planning, and situated natural language understanding.



Peter N. Prokopowicz is a post-doctoral fellow in the AI Laboratory at the University of Chicago. He is currently working on the development of a new software platform for active vision in mobile robots. He received his Ph.D. from the Institute for the Learning Sciences at Northwestern University and his M.S. and B.S. from the University of Michigan. His thesis presented a computational neural model of perceptual integration across eye movements. He was a member of the technical staff at AT&T Bell Laboratories for several years. Recently, he cofounded Perceptual Robotics, Inc., which develops custom Internet-based remote video technology, and the Electronic Community, Inc., which provides the NETHOMES World Wide Web publishing service.



Michael J. Swain received his Ph.D. in computer science from the University of Rochester in 1990. He is now assistant professor of computer science at the University of Chicago. Swain's current research interests include active vision, gesture recognition, and content-based indexing into image and video databases. Swain organized the 1991 National Science Foundation Workshop on Active Vision and edited the *International Journal of Computer Vision* special issues on active vision in 1993 and 1994.



Roger E. Kahn is a graduate student in the AI Lab at the University of Chicago. He attended Kalamazoo College from 1987 to 1991 and studied at the Eotvos Lorand University on the Budapest Semesters in Mathematics program. His current research involves developing PERSEUS, a real-time robot vision platform, and using it to recognize gestures.



David Franklin received his B.S. in computer science from the University of Washington in 1994. He is currently a Ph.D. student at the University of Chicago, where his current research interests include autonomous mobile robotics and the representation of action.