Happy Patrons Make Better Tippers
Creating a Robot Waiter Using Perseus and the Animate Agent Architecture [*]

David Franklin, Roger E. Kahn, Joshua Flachsbart, Michael J. Swain and **R. James Firby**
Department of Computer Science
University of Chicago
{franklin, kahn, josh, swain, firby}@cs.uchicago.edu

Abstract

As autonomous robots become increasingly adept at performing simple tasks like moving from place to place and picking up and delivering objects, it is becoming apparent that an important area of robotic research is that of developing natural interfaces for controlling them. In the context of building a robot "waiter", we demonstrate the use of the Perseus architecture for gesture recognition, teamed with the Animate Agent architecture for tightly coupled perception and action. Of particular significance is the ease of implementing this task utilizing the architectures and routines we have already created for other tasks.

1 Introduction

Designing natural and convenient human-machine interfaces is an important and never-ending struggle for designers of technology. One of the most challenging and interesting domains in which to attempt this task is in the context of human / mobile-robot interaction, where giving the robot as many human-like capabilities as possible makes for the most natural interface.

In previous work, we developed the Perseus visual architecture and demonstrated the use of Perseus in recognizing and interpreting human pointing gestures (Kahn *et al.* 1996)(Kahn & Swain 1995). The design of the system was to develop re-usable visual representations and operators for creating them, that could be used in a number of gesture recognition tasks. The system was also designed to make maximum use of context such as might be supplied by a reactive control system, as described below.

In related work, we have designed an architecture for tightly-coupled perception and action, called the Animate Agent architeture, and demonstrated its use in a "clean up the office" task that has been the focus of successive robot competitions (Firby *et al.* 1995)(Firby *et al.* 1996). In the clean-up task, the robot is let loose in a small room with soda cans and paper cups which are to be picked up and sorted into trash and recycling bins.

This paper describes work on a research project that utilizes gesture recognition to enable our robot to serve as a waiter in a simplified restaurant domain. The robot observes the restaurant patron, waits until he performs a gesture indicating that he wants the robot to do something, and then attempts to make the patron happy by meeting his needs. The robot waiter is able to:

- Perform visual identification of gestures that are interpreted to mean the requesting of or offering of an object.
- Visually track a person and locate his feet.
- Safely navigate to a person.
- Physically hand objects to and receive objects from a person.

This research demonstrates the flexibility and extensibility of our current work on the Perseus visual architecture and the Animate Agent architecture, as we were quickly (within two months) able to program our robot to accomplish a new task. As we had hoped in previous papers (???), the new task allowed a great deal of our old routines to be utilized and required a minimal amount of additional programming.

2 Perseus

Perseus(Kahn *et al.* 1996)(Kahn & Swain 1995) is a purposive visual architecture that has been used to recognize gestures performed by people in non-engineered environments. Knowledge about the task and environment is used at all stages of processing to best interpret the scene for the current situation. Once the

[*] Many thanks to the others who contributed to this work, including Peter Prokopowiz and the students of CS 359: Shannon Bradshaw, Mazin As-Sanie, Mark Langston, Jiayu Li and Alain Roy. This work was supported in part by ONR contract N00014-93-1-0332 and by ARPA contract N00014-93-1-1394.

visual operators are chosen, contextual knowledge is used to tune them for maximal performance. Redundant interpretation of the scene provides robustness to errors in interpretation. Fusion of independent types of information results in increased tolerance when assumptions about the environment fail. And windows of attention are used to improve speed and remove distractions from the scene. Reuse is a major issue in the design of Perseus, information about the environment and task is explicitly represented so it can easily be used in a variety of tasks. A clean interface to Perseus is provided for symbolic higher level systems(Firby *et al.* 1995) like the RAP reactive execution system(Firby 1989).

The Perseus system is composed of six types of components: feature maps, object representations(ORs), markers(Chapman 1991), visual routines(Ullman 1984), a segmentation map, and a long term visual memory(LTVM). A reactive execution system(Firby 1989) interfaces to Perseus by calling visual routines(Ullman 1984)(Chapman 1991). Visual routines are the top level structures in the Perseus system, addressing complex visual tasks like waiting for a person to enter the scene or finding the area a person points to. When visual routines are called, they are passed parameters. These parameters may include symbols used by the higher level system to reason about objects in the world like a person or a soda can.

The LTVM provides the translation between symbols used by the higher level system and the data needed to represent real-world objects. Visual routines use the LTVM to find an OR for the symbols passed to them. They then use ORs to understand which objects are in the scene and what the properties of those objects are.

ORs are encapsulations of data about a particular object and methods for examining the object represented. When a method is called, it returns information that the visual routine may use to understand the scene. Typical methods locate a person's hands, or find a soda can in a specified area. The methods of an OR have access to higher level information so they can select different operators for interpreting the scene depending on the situation. ORs are not static representations, instead the data they contain can be modified by their methods. For instance, a method for locating a person may record the color of the person's clothing so he can more easily be located later.

Segmentation is often facilitated by knowing which points in the scene are *not* part of the object. This information can be known if other ORs have performed segmentations. To support this, a global segmentation map is maintained. Each time an OR segments an object, the segmentation map is updated. This map may then be used to decide where to look or where *not* to look.

When an OR needs to track something, it instantiates a marker(Chapman 1991) and parameterizes the marker with a tracking function. Visual routines and ORs may then query the marker when they need to know where the object is. Markers track objects until either they lose them or a signal is received explicitly saying to stop tracking.

ORs and markers base their computations on a set of feature maps rather than directly upon the frames grabbed from the camera. Feature maps are retinotopic interpretations of the scene detecting the presence of edges, motion, and other local properties. They are parameterized by ORs and markers based on higher level information about the environment and goals of the agent to more effectively recognize features of interest.

3 The Animate Agent architecture

Our robot is controlled by the combination of a hierarchical reactive planner (the RAPs system) and a low-level system of skills (the CRL system.) Together, the two components make up the Animate Agent architecture (Firby *et al.* 1995).

The CRL system (Chicago Robot Language) allows for the parallel execution of low-level skills. These facilitate control of the robot's actuators and provide an interface with the sensing systems of the robot (including Perseus.) Skills are enabled in sets which communicate with each other by setting an reading global channel values, allowing for the creation of tight servo loops for such activities as tracking and navigating.

The RAP system (Reactive Action Package) provides an interpreter for "sketchy" plans – plans where some steps must be determined at run-time. It maintains a memory containing what is believed to be true about the world, which is used to determine what method to use for a plan. The RAP system also configures and controls the CRL system, enabling sets of skills to be run in parallel. The CRL system communicates with the RAP system by sending signals that describe significant changes in state (for example "I have arrived at the goal," "My hand is now empty," or "I can no longer find the object I was tracking.") These signals give the RAP system the information it needs to monitor the progress of its plans.

4 Recognizing service requests

In the context of the robotic waiter task, two requests can be made of the waiter. The patron may request either that a soda be delivered to him or that an empty soda be taken from him. The patron communicates these requests by gesturing to the robot: a patron extending an empty hand is understood to be requesting a soda can and a patron extending a hand holding a soda can is understood to be requesting that the can be taken away.

4.1 Locating and tracking the patron

The first step in the task involves determining when a restaurant patron has entered the area and then observing him until he gestures that the robot ought to do something. Determining when someone has entered the restaurant is accomplished by telling the Perseus system to instantiate an OR for the person to be waited upon (using the `wait-for-patron-request` visual routine.) The OR is a generic instance of a person OR – one that contains no details about such things as the person's body color or position. This OR has a `locate` method which is used to find a person in the scene(Kahn *et al.* 1996). This method assumes that people are the only objects that move in the scene and waits for a region of motion. When the `locate` method is successful, it means that a patron has been located and can be tracked.

Once the patron is found, the person OR's `segment` method(Kahn *et al.* 1996) is called to segment him from the rest of the scene, locate his body parts, and extract the colors of his body. The initial segmentation of the person is done by first deciding where *not* to look. ORs for the lights, floor, and background(Kahn *et al.* 1996) are instantiated which update Perseus' segmentation map with the pixels corresponding to these objects. Once the non-interesting regions of the scene have been identified, the connected components in the resulting regions are combined to produce a segmentation of the person.

The color histogram of the segmentation is stored in the person OR. A marker is placed on the centroid of the segmentation. This marker parameterizes the color feature map to backproject(Swain & Ballard 1991) the person's body colors onto the scene. The body is tracked as the peak in this map, ignoring regions marked as background, floor, or lights in the segmentation map.

Next, the top, extreme left, extreme right, and bottom of the segmentation are marked as the head, left hand, right hand, and feet respectively. Each of these markers are segmented as a small region around their current locations. As they are tracked, they reposition themselves on the top, left, right, or bottom of that segmentation using constraints on how far they can be from each other relative to the person's height.

4.2 Observing the patron's gesture

Once the person has been located and his body parts are being tracked the `wait-for-patron-request` visual routine monitors the hand positions until one is held stationary and out from the patron's side for a short time. After this gesture occurs the person's hand is examined to see whether it is holding anything. This is determined by computing what percentage of the hand pixels are "skin colored." If few of the pixels are skin colored, then the hand must be holding something that occludes the hand and conversely, if there is something sufficiently large in the hand, it will occlude much of the hand. A threshold on the percentage of pixels in the hand segmentation is used to determine whether or not there is something in the hand.

The method we use to identify skin colored pixels is based on Buluswar's work on non-parametric classification of pixels under varying illumination(Buluswar & Draper 1994). A multivariate decision tree (MDT) algorithm(Draper, Brodley, & Utgoff 1994) is used to separate the skin colored region of the color space from the rest. Training data was taken from hand cropped images. Figures 4.2 and 4.2 show the skin detection results on two images.

Figure 1: Recognizing a hand holding a can.

Figure 2: Recognizing a hand without a can.

The task of the multivariate decision tree algorithm is to find planes in RGB space that separate skin colors from other colors. It is assumed that skin color is contained within a convex shape in RGB space that contains few, if any, non-skin colors. Given sets of skin-colored and non-skin-colored RGB values, the MDT algorithm uses the recursive least squares method of error reduction to place the multivariate planes dividing the color space. Our iterative training technique was to take pixels that were misclassified by the MDT produced by the previous iteration and use them as the next iteration's training data. Most of the misclassified pixels had color values lying close to the region boundary and so provided a more accurate definition of the boundary.

After many iterations of computing the skin color MDT, a accurate definition of the skin-colored region in RGB space was found. To facilitate fast identification of skin-colored pixels, the information stored in the MDT was transferred to a lookup table, a binary array that maps RGB color values into a determination of whether or not they are skin colored.

Perseus' color feature map(Kahn *et al.* 1996) was extended so that in addition to performing color histogram backprojection, it could accept a color lookup table as an argument and perform backprojection from that table. To see if the person's hand is empty the `wait-for-patron-request` visual routine parameterizes the color feature map with the skin lookup table and examines the map in a small region surrounding the marker on the person's hand that is gesturing. If more than 1/3 of the pixels in the person's segmentation around the hand are skin-colored, the hand is considered to be empty. Otherwise it is considered to be holding an object.

5 Preparing to serve the patron

Once the patron has requested service and the system has determined what type of service the patron is requesting, the robot must service that request. This involves either locating a can and delivering it to the patron (the patron is requesting something) or approaching the person and taking a can from him.

5.1 Finding an object for the patron

When the patron requests an object, the robot assumes that he wants a soda can and so it must find a soda can to give him. If the robot is already holding a soda can, then it will give that can to the patron. Otherwise, the robot must look around for one.

The robot repeatedly selects areas on the floor that it has not looked at yet, pans its cameras to look at them, and uses the Perseus system to detect and locate all soda cans in its field of view. Once the robot has located a can, it will move to the can and visually align with it. Finally, the robot lowers its arm, rolls forward until the can is in its gripper, stops, and closes its gripper around the can. With the can in its gripper, the robot is then ready to fold up its arm and deliver the soda can to the patron. For a detailed description of how the robot locates and picks up cans (used in previous years' IJCAI robot competitions,) see (Firby *et al.* 1995).

5.2 Visually approaching the patron

Once the patron's needs have been determined through the gesture recognition (and the robot has picked up a soda can, if necessary,) the robot must approach the patron in order to serve him. The patron's location is determined by converting the screen coordinates of the person OR's feet marker into floor coordinates using knowledge of the position and orientation of the robot's camera.

The person OR's feet marker tracks the feet by finding the lowest pixel in a segmentation of the person. This segmentation is usually computed by combining the segmentations of the floor, lights, and background with the color histogram backprojection of the person's body. When the robot is in motion, however, the background representation ceases to function because the background is no longer stable(Kahn *et al.* 1996) from the robot's perspective. Consequently, only the color backprojection is used for segmentation while the robot is in motion.

5.2.1 Color histogram backprojection

The visual tracking of the patron by color makes use of the technique of color histogram backprojection (Swain & Ballard 1991). In this technique, each possible color value (in a given color space) has a corresponding bin in the histogram. The value of each bin represents the number of pixels in a segmented image that have that color value.

Color histogram backprojection requires two color histograms: one for the object being searched for and one for the background. For each pixel in the image to be backprojected, the bin values for that pixel's value are retrieved from the object and background histograms. The corresponding pixel in the backprojection image is then set to the normalized ratio of the object value to the background value. This technique causes color values unique to either the object or background to take extreme values and color values common to both to take moderate values and allows for accurate segmentation of objects in many situations.

The color histogram of the patron is acquired using Perseus to get a segmentation of the patron and then computing the color histogram of the segmented region. We use an 8-bit HSV color space that is biased heavily towards hue. Due to the relative lack of attention to saturation and value (intensity), the patron can be tracked across dramatic changes in lighting.

5.2.2 Visually tracking the patron's feet

To locate the patron's feet in a color image, given color histograms for the patron and the background, we first approximate the centroid of the patron by computing a backprojected image of the patron, blurring it using a large kernel and finding the peak value in the resulting image. Then, we look for the lowest row with sufficient histogram response in the region defined by a rectangle extending from the peak down to the bottom of the image and wide enough to be certain to contain at least one of the patron's feet. (This is how we find the lowest row in the image that contains part of the patron.) Finally, the x-coordinate of the peak and the y-coordinate of the lowest row are used to define the location of the patron's feet.

5.2.3 Approaching the patron

The image coordinates of the patron's feet, combined with a known position of the camera, can be used to compute the floor coordinates of the patron's feet. These floor coordinates are used as the goal for the robot's navigation system and the robot adjusts its pan-tilt head so that the floor coordinates are centered in the camera's field of view. Actually approaching the patron then becomes a cycle of locating the patron's feet, updating the current navigation goal and redirecting the cameras. This cycle is accomplished through the interaction of three skills:

`track-feet` This skill communicates with the Perseus system, taking the latest screen coordinates of the patron's feet, converting the screen coordinates into world coordinates, and setting the `target-x` and `target-y` channels of the CRL system to these values.

`pan-to-target` This skill reads the target channels and positions the pan-tilt head such that the target is centered in the lower half of the camera's field of view (given the robot's current position.)

`move-and-avoid` This skill drives the robot towards the target while using its sonars to detect and avoid obstacles along the way. When the robot gets sufficiently close to the target, it sends a message to the RAP system indicating that it has reached the target.

The RAP system enables all three skills in parallel and then waits for the `move-and-avoid` skill to send a message back. The `pan-to-target` and `move-and-avoid` skills work independently of what is being tracked. In fact, these skills are also used in other tasks such as visually approaching a pop can to be picked up.

6 Exchanging objects with the patron

Having our robot physically interact with people has proven to be quite tricky: it does not have the complicated sensing ability of a human and its hand and arm have limited degrees of freedom and move very slowly. As a result, we rely on the cooperation of the patron to facilitate the exchange. When the robot offers an object, the patron must remove it from the robot's extended arm, and when the robot accepts an object, the patron must hold the object in the robot's gripper until it is securely held in the robot's grasp. Despite the fact that the robot does not handle exchanges in quite the same way as people do, people seem to have no problems performing the exchanges with the robot and interpret the interaction as being natural.

Figure 3: The robot offering a can to a patron.

6.1 Orienting to the patron

Before performing an exchange with the patron, the robot must orient itself to him. RAP memory holds information about where the patron is standing (asserted after the patron has been approached) and which hand he gestured with (asserted when the patron makes his gesture.) To orient with the patron, the RAP system uses the information it has stored to estimate the location of the patron's hand and then enables a skill to rotate the robot to face that location.

6.2 Offering an object to the patron

To offer an object to the patron, the robot extends its arm upward and outward to its highest position and tells the patron to take the object out of its hand.

Then, a skill is enabled that monitors whether there is anything in the robot's gripper: the robot can detect when an infrared beam running between the fingers of the gripper has been broken. When the skill detects that the beam is no longer broken, it signals the RAP system that the gripper is no longer holding anything. This is the same skill that is used to make sure that the robot does not drop objects while it is carrying them from place to place.

While this skill is running, the RAP system keeps track of how long it is waiting for the patron to take the object. If the patron delays, the robot will complain and if he waits too long the robot will give up: the grasping skill will be disabled, the arm lowered, and the robot will deposit the can in the nearest recycling bin. Figure 3 shows the robot offering a can to a patron.

6.3 Accepting an object from the patron

To accept an object from the patron, the robot extends its arm as before, opens its gripper, and tells the patron to place the object in its gripper. Then, a skill is enabled that waits for the beam in the gripper to be broken (indicating that something is in the gripper), closes the gripper, and sends the RAP system a signal indicating that the robot has grasped an object. This is the same skill that is used to pick up small objects from the floor (to pick up a small object, we have the robot visually orient to the object, roll forward until the beam in the gripper is broken, and then stop and grasp the can.) If the patron takes too long, the robot will give up and return to its initial location.

7 Related work

The work of Kortenkamp *et al* (Kortenkamp, Huber, & Bonasso 1996) shows a number of similarities in their approach to designing a system for human-robot interaction, in part because their system is based upon some of the technologies developed for the Animate Agent Project, such as the RAP system and the three-level control system that underlies the Animate Agent architecture (Bonasso *et al.* To appear 1996).

8 Future work

There are two areas of related work that will directly extend the work described in this paper. One is work on situated natural language interpretation, which interprets utterances in relation to the context provided by the perceptual systems and RAP memory. The other is Gargoyle, an environment we are developing for situated real-time computer vision (Prokopowicz *et al.* 1996). Gargoyle will permit the RAP system to assemble real-time visual routines on-the-fly. Its multithreaded interpreter is cross-platform, and is designed to run over the Teleos AVP-100 system for computing correspondence for stereo and motion. It will permit us to port Perseus to an on-board multi-processor.

9 Conclusion

This paper describes research that extended previous work using the Perseus visual architecture and the Animate Agent architecture, enabling our mobile robot to accomplish a new task, that of serving as a robot waiter in a simple restaurant environment. The strengths of both of these architectures was demonstrated as a great deal of the original functionality was able to be used in new ways and a minimal amount of additional programming proved necessary. It is our hope, and expectation, that future tasks can be implemented with similar ease, utilizing the tools we have already developed.

References

Bonasso, R. P.; Gat, E.; Firby, R. J.; Kortencamp, D.; Miller, D. P.; and Slack, M. G. To appear, 1996. Experiences with an architecture for intelligent, reactive agents. *Journal of Experimental and Theoretical Artificial Intelligence.*

Buluswar, S., and Draper, B. 1994. Nonparametric classification of pixels under varying outdoor illumination. *Image Understanding Workshop.*

Chapman, D. 1991. *Vision, Instruction, and Action.* MIT Press.

Draper, B.; Brodley, C.; and Utgoff, P. 1994. Goal-directed classification using linear machine decision trees. *Machine Learning.*

Firby, R. J.; Kahn, R. E.; Prokopowicz, P. N.; and Swain, M. J. 1995. An architecture for vision and action. *International Joint Conference on Artificial Intelligence.*

Firby, R. J.; Prokopowicz, P. N.; Swain, M. J.; Kahn, R. E.; and Franklin, D. 1996. Programming CHIP for the IJCAI-95 robot competition. *AI Magazine* 17(1).

Firby, R. J. 1989. *Adaptive Execution in Complex Dynamic Worlds.* Ph.D. Dissertation, Yale.

Kahn, R. E., and Swain, M. J. 1995. Understanding people pointing: The Perseus system. *International Symposium on Computer Vision.*

Kahn, R. E.; Swain, M. J.; Prokopowicz, P. N.; and Firby, R. J. 1996. Gesture recognition using the perseus architecture. *Computer Vision and Pattern Recognition.*

Kortenkamp, D.; Huber, E.; and Bonasso, R. P. 1996. Recognizing and interpreting gestures on a mobile robot. *American Association for Artificial Intelligence.*

Prokopowicz, P. N.; Swain, M. M.; Firby, R. J.; and Kahn, R. E. 1996. Gargoyle: An environment for real-time, context-sensitive active vision. *American Association for Artifical Intelligence.*

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

Ullman, S. 1984. Visual routines. *Cognition* 18:97–159.