Recognizing and interpreting gestures on a mobile robot

David Kortenkamp, Eric Huber, and R. Peter Bonasso
Metrica, Inc.
Robotics and Automation Group
NASA Johnson Space Center – ER2
Houston, TX 77058
korten@mickey.jsc.nasa.gov

Abstract

Gesture recognition is an important skill for robots that work closely with humans. Gestures help to clarify spoken commands and are a compact means of relaying geometric information. We have developed a real-time, three-dimensional gesture recognition system that resides on-board a mobile robot. Using a coarse three-dimensional model of a human to guide stereo measurements of body parts, the system is capable of recognizing six distinct gestures made by an unadorned human in an unaltered environment. An active vision approach focuses the vision system's attention on small, moving areas of space to allow for frame rate processing even when the person and/or the robot are moving. This paper describes the gesture recognition system, including the coarse model and the active vision approach. This paper also describes how the gesture recognition system is integrated with an intelligent control architecture to allow for complex gesture interpretation and complex robot action. Results from experiments with an actual mobile robot are given.

Introduction

In order to work effectively with humans, robots will need to track and recognize human gestures. Gestures are an integral part of communication. They provide clarity in situations where speech is ambiguous or noisy (Cassell 1995). Gestures are also a compact means of relaying geometric information. For example, in robotics, gestures can tell the robot where to go, where to look and when to stop. We have implemented a real-time, three-dimensional gesture recognition system on a mobile robot. Our robot uses a stereo vision system to recognize natural gestures such as pointing and hand signals and then interprets these gestures within the context of an intelligent agent architecture. The entire system is contained on-board the mobile robot, tracks gestures at frame-rate (30 hz), and identifies gestures in three-dimensional space at speeds natural to a human.

Gestures for mobile robots

Gesture recognition is especially valuable in mobile robot applications for several reasons. First, it provides a redundant form of communication between the user and the robot. For example, the user may say "Halt" at the same time that they are giving a halting gesture. The robot need only recognize one of the two commands, which is crucial in situations where speech may be garbled or drowned out (e.g., in space, underwater, on the battlefield). Second, gestures are an easy way to give geometrical information to the robot. Rather than give coordinates to where the robot should move, the user can simply point to a spot on the floor. Or, rather than try to describe which of many objects the user wants the robot to grasp, they can simply point. Finally, gestures allow for more effective communication when combined with speech recognition by grounding words such as "there" and "it" with recognizable objects.

However, mobile robot applications of gesture recognition impose several difficult requirements on the system. First, the gesture recognition system needs to be small enough to fit onto the mobile robot. This means that processing power is limited and care must be taken to design efficient algorithms. Second, the system needs to work when the robot and the user are moving, when the precise location of either is unknown and when the user may be at different distances from the robot. It is also likely that objects will be moving in the background of the image. Third, precise calibration of cameras is difficult if not impossible on a mobile platform that is accelerating and decelerating as it moves around. Finally, the system needs to work at a speed that is comfortable for human tasks, for example the halting gesture needs to be recognized quickly enough to halt the robot within a reasonable time. In this paper we present a gesture recognition system that meets these requirements.

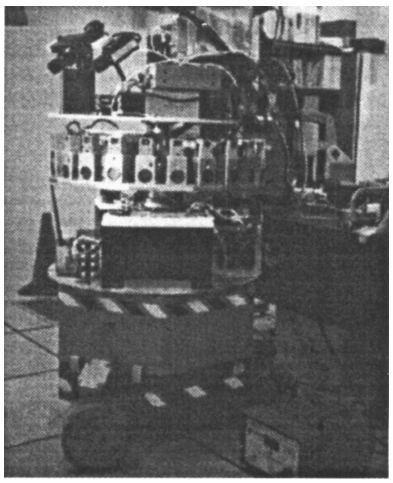

Figure 1: A mobile robot with on-board vision system.

Related work

Gesture recognition has become a very important research area in recent years and there are several mature implementations. The ALIVE system (Darrell *et al.* 1994) allows people to interact with virtual agents via gestures. The ALIVE system differs from ours in that the cameras are fixed (i.e., not on a mobile platform as ours are) and that it requires a known background. Similar restrictions hold for a system by Gavrila and Davis (Gavrila & Davis 1995). The Perseus system (Kahn *et al.* 1996) uses a variety of techniques (e.g., motion, color, edge detection) to segment a person and their body parts. Based on this segmentation the Perseus system can detect pointing vectors. This system is very similar to ours in that it is mounted on a mobile robot and integrated with robot tasks. The Perseus system differs from ours in that it requires a static background, doesn't detect gestures in three dimensions and relies on off-board computation, which can cause delays in recognition of gestures. Wilson and Bobick (Wilson & Bobick 1995) use a hidden Markov model to learn repeatable patterns of human gestures. Their system differs from ours in that it requires people to maintain strict constraints on their orientation with respect to the cameras.

Recognizing gestures

Our gesture recognition system consists of a stereo pair of black and white cameras mounted on a pan/tilt/verge head that is, in turn, mounted on the mobile robot (see Figure 1). The basis of our stereo vision work is the PRISM-3 system developed by Keith Nishihara (Nishihara 1984). The PRISM-3 system provides us with low-level spatial and temporal disparity measurements. We use these measurements as input to our algorithms for gesture recognition.

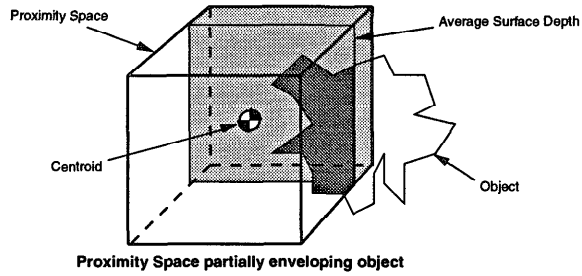

Figure 2: A proximity space.

Our gesture recognition process has two components. First, we concentrate our vision system's attention on small regions of 3-D visual space. We call these regions *proximity spaces*. These spaces are designed to react to the visual input much as a robot reacts to its sensory input. Second, we spawn multiple proximity spaces that attach themselves to various parts of the agent and are guided by a coarse, three-dimensional model of the agent. The relationships among these proximity spaces gives rise to gesture recognition. Each of these two components is described in the following two subsections and then their application to gesture recognition.

The proximity space method

The active vision philosophy emphasizes concentrating measurements where they are most needed. An important aspect of this approach is that it helps limit the number of measurements necessary while remaining attentive to artifacts in the environment most significant to the task at hand. We abide by this philosophy by limiting all our measurements in and about cubic volumes of space called proximity spaces.

Within the bounds of a proximity space, an array of stereo and motion measurements are made in order to determine which regions of the space (measurement cells) are occupied by significant proportions of surface material, and what the spatial and temporal disparities are within those regions. Surface material is identified by detecting visual texture, i.e., variations in pixel values across regions of the LOG (Laplacian of Gaussian) filtered stereo pair (see Figure 2).

The location of a proximity space is controlled by behaviors generating vectors that influence its motion from frame to frame. Behaviors generate motion vectors by assessing the visual information within the proximity space. There are behaviors for following, clinging, pulling, migrating to a boundary and resiz-

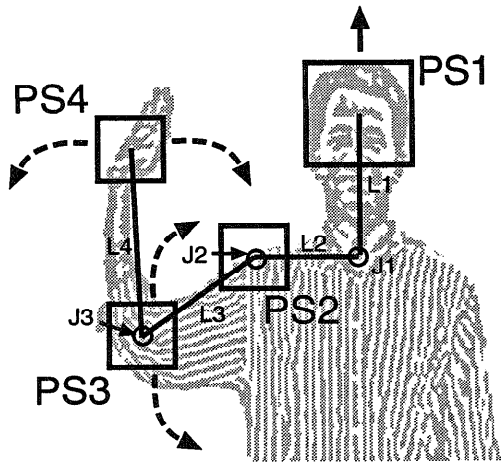

Figure 3: Coarse 3-D model of a human used for gesture recognition.

ing (which does not generate a motion vector, but a size for the proximity space). Patterned after the subsumption architecture (Brooks 1986), these behaviors compete for control of the proximity space. In dynamic terms, the proximity space acts as an inertial mass and the behaviors as forces acting to accelerate that mass (see (Huber & Kortenkamp 1995) for a more detailed description).

Chaining multiple proximity spaces using a human model

In order to recognize gestures, multiple proximity spaces are spawned, which attach themselves to various body parts in the image of the gesturing person. Each of these proximity spaces has its own set of behaviors independently controlling its location in space. However, these behaviors are constrained by a coarse three-dimensional, kinematic model of a human that limits their range of motion. With perfect tracking there would be no need for a model as the proximity spaces would track the body parts in an unconstrained manner. However, real-world noise may sometimes cause the proximity space to wander off of their body parts and begin tracking something in the background or another part on the body. While the behaviors acting on the proximity spaces continue to generate motion vectors independent of the model, the final movement of the proximity spaces is overridden by the model if the generated vectors are not consistent with the model.

We have chosen a model that resembles the human skeleton, with similar limitations on joint motion. The model, shown in Figure 3, consists of a head connected to two shoulder joints. There are four proximity spaces that are tracking various parts of the body. The first proximity space (PS1) is tracking the head. Its behaviors allow it to move freely, but are also biased to migrate it upward. The second proximity space (PS2) tracks the shoulder. PS1 and PS2 are connected by two links, L1 and L2. These links are stiff springs that allow very little movement along their axes. The lengths of L1 and L2 are set at the start of tracking and do not change during tracking. We have a procedure for automatically determining the lengths of the links from the height of the person being tracked, which is described later in the paper. The connection between L1 and L2 (J1) acts as a ball joint. The motion of L2 relative to L1 (and thus, PS2 relative to PS1) is constrained by this joint to be 0 deg in the up-down direction (i.e., shrugging of the shoulders will not affect the location of the proximity space). The joint J1 also constrains the movements of L2 relative to L1 by ± 38 deg in the in-out direction (i.e., towards and away from the camera). This means that the person must be facing the cameras to within 38 deg for the gesture recognition system to work correctly.

The third proximity space (PS3) is tracking the end of the upper arm and is connected to PS2 by the link L3. Again, L3 is modeled as a very stiff spring with a length fixed at the start of tracking. L3 is connected to L2 with a ball joint (J2). L3 can move relative to L2 up at most 75 deg and down at most 70 deg. It can move into or out-of perpendicular to the camera by at most ± 45 deg. This means that a pointing gesture that is towards or away from the robot by more than 45 deg will not be recognized. Finally, the fourth proximity space (PS4) tracks the end of the lower arm (essentially the hand) and is connected to PS3 with link L4 at joint J3. The range of motion of L4 relative to L3 at J3 is up at most 110 deg, down at most -10 deg and into or out-of perpendicular with the camera by at most ± 45 deg. All of the links in the model are continuously scaled based on the person's distance from the cameras.

One limitation of our gesture recognition system is that it can only track one arm at a time. There are two reasons for this limitation. First, we do not have enough processing power to calculate the locations of seven proximity spaces at frame rate. Second, when both arms are fully extended the hands fall out of the field of view of the cameras. If the person backs up to fit both hands into the field of view of the cameras then the pixel regions of arms are too small for tracking. The arm to be tracked can be specified at the start of the tracking process and can be switched by the user. While Figure 3 only shows the model of the right arm for simplicity, the model for the left arm is simply a

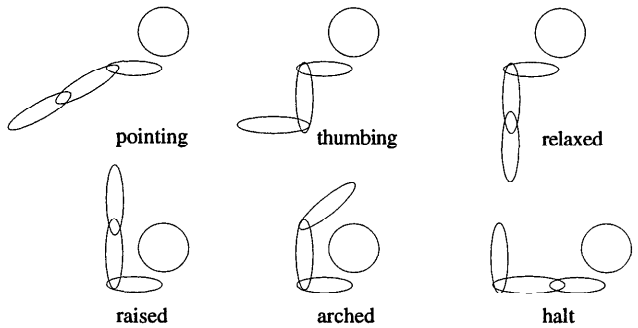

Figure 4: Recognizable gestures.

mirror image of the right.

Experiments, which are described in more detail later, show that our system can recognize gestures from distances as close as 1.25 meters from the camera (at which point the person's arm extends out of the field of view of the cameras) to as far as 5 meters away from the robot. The system can track a fully extended arm as it moves at speeds up to approximately 36 deg per second (i.e., a person can move their arm in an arc from straight up to straight down in about five seconds without the system losing track).

Acquisition and reacquistion using the model Gesture recognition is initiated by giving the system a camera-to-person starting distance and a starting arm. Four proximity spaces are spawned and lay dormant waiting for some texture to which to attach themselves. As a person steps into the camera at approximately the starting distance the head proximity space will attach itself to the person and begin migrating towards the top, stopping when it reaches the boundary of the person. The other three proximity spaces are pulled by their links up along with the head. While the person's arm is at their side, these proximity spaces are continually sweeping arcs along the dashed arrows shown in Figure 3 looking for texture to which to attach themselves. When the arm is extended the three proximity spaces "lock onto" the arm and begin tracking it. If they lose track (e.g., the arm moves to fast or is occluded) they begin searching again along the dashed arcs shown in Figure 3. If the head proximity space loses track it begins an active search starting at the last known location of the head and spiraling outward. Many times this re-acquisition process works so quickly that the user never realizes that tracking was lost.

Defining gestures

Figure 4 shows the gestures that are currently recognized by the system. These gestures are very easily determined by looking at the relative angles between the links L2, L3 and L4 at the joints J2 and J3 (see Figure 3). Let's call the angle between L2 and L3 (i.e., the upper arm angle) $\Theta 1$ and the angle between L3 and L4 (i.e., the lower arm angle) $\Theta 2$. 0 deg is straight out to the left or right. Then, if $\Theta 1 < -50$ deg and $\Theta 2 < 45$ deg the gesture is *relaxed.* If $\Theta 1 < -50$ deg and $\Theta 2 > 45$ deg the gesture is *thumbing.* If -50 deg $< \Theta 1 < 45$ deg and $\Theta 2 < 45$ deg the gesture is *pointing.* If -50 deg $< \Theta 1 < 45$ deg and $\Theta 2 > 45$ deg the gesture is *halt.* If $\Theta 1 > 45$ deg and $\Theta 2 < 45$ deg the gesture is *raised.* If $\Theta 1 > 45$ deg and $\Theta 2 > 45$ deg the gesture is *arched.* Thus, the person is always producing some kind of gesture based on the joint angles.

Gesture recognition is not immediate as that may lead to many spurious gestures. Confidence in a gesture is built up logarithmically over time as the angles stay within the limits for the gesture. When the logarithmic confidence passes a threshold (0.8 in our experiments) then the gesture is reported by the system. That gesture continues to be reported until the confidence drops below the threshold.

This gesture recognition technique does not currently support recognizing gestures that occur over time (e.g., a waving gesture). We believe that our approach, because it is active, lends itself to this and we are working towards implementing it.

Connecting gestures to robot action

Simply recognizing gestures is not enough for them to be useful; they need to be connected to a specific robot actions. For the last several years we have been working on an intelligent control architecture, known as 3T, which can integrate reactive vision and robot processes with more deliberative reasoning techniques to produce intelligent, reactive robot behavior (Bonasso *et al.* 1995). The architecture consists of three layers of control: skills, sequencing and planning. Only the first two layers (skills and sequencing) have been used in the system described in this paper. The next two subsections will describe the skills of our robot and how those skills can be intelligently sequenced to perform tasks.

Visual skills

Skills are the way in which the robot interacts with the world. They are tight loops of sensing and acting that seek to achieve or maintain some state. Skills can be enabled or disabled depending on the situation and the set of enabled skills forms a network in which information passes from skill to skill. Figure 5 shows the skill network for our work in gesture recognition. The

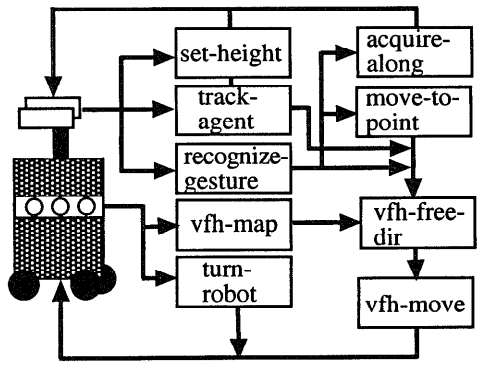

Figure 5: Mobile robot skills for gesture recognition.

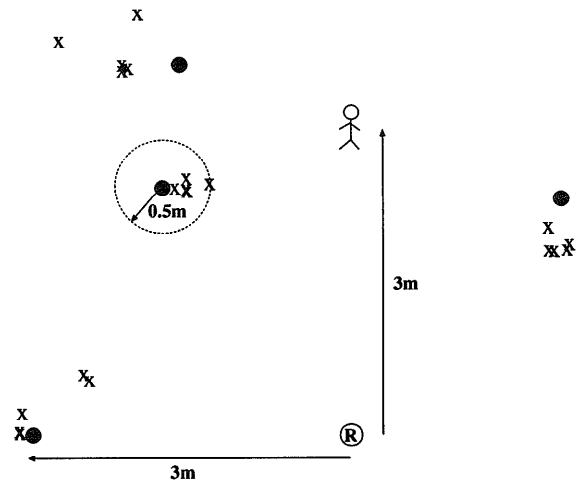

Figure 6: A sample of experimental results. The person is standing directly in front of the robot and pointing at different points on the floor (black circles). The 'X' is the point that the robot calculated as the intersection between the pointing gesture and the floor.

skills labeled vfh are obstacle avoidance and robot motion skills base on the Vector Field Histogram method (Borenstein & Koren 1991). They take a goal location, generated from any skill, and move the robot to that goal location. The move-to-point, the track-agent and the recognize-gesture skills allow can provide goal locations to the vfh skills.

The recognize-gesture skill encapsulates the processes described in the previous section and produces one of the five gestures or no gesture as output. It also generates as output the (x,y,z) locations of the four proximity spaces when the gesture was recognized. The next several subsections described the more interesting gesture recognition skills in detail.

Moving to a point This skill produces an (x,y) goal for the robot corresponding to the location at which the person is pointing. This skill takes the (x,y,z) location of the centroid of the shoulder proximity space (PS2 in Figure 3) and the hand proximity space (PS4 in Figure 3) and computes a three-dimensional vector. It then determines the intersection point of this vector with the floor. Assuming the vector does intersect with the floor, the skill begins generating a goal for the robot and the motion control and obstacle avoidance skills move the robot to that point.

We conducted a number of experiments to determine the accuracy of the pointing gesture. The experimental set-up was to have a person point to a marked point on the floor. The vision system would recognize the pointing gesture and the move-to-point skill would determine the intersection of the pointing vector with the floor. We would then compare this point with the actual location of the target. We choose eight different target points on the floor in various directions and at various distances. We pointed five times at each target point. Two different people did the point, both of them familiar with the system. No feedback was given to the user between trials. Figure 6 shows a sample of those points and the system's performance.

For the five points that were between 2.5 and 4.5 meters away from the person, the mean error distance from the target to the vector intersection was 0.41 meters, with a standard deviation of 0.17. As the distance from the person to the target grew the error also grew rapidly, up to a mean error of over 3 meters at 5.5 meters away.

These results need to be taken with a grain of salt. There are several factors that can introduce errors into the system and that cannot be accounted for, including: how accurately a person can actually point at a spot; the initial accuracy of the robot both in position and orientation; and the tilt of the robot due to an uneven floor.

Acquiring along a vector When this skill is enabled, a pointing gesture will result in the vision system searching along the pointing vector and stopping if it acquires a distinct object. The vision system then begins tracking that object. This skill takes the (x,y,z) location of the centroid of the shoulder proximity space and the hand proximity and computes a three dimensional vector. The skill then causes the vision system to search along through a tube of space surrounding that vector until a patch of significant texture is encountered. The skill stops searching after a certain distance, which is passed to the skill as a parameter at run time. Informal experiments allowed two people

standing about 2 meters apart to "pass control" of the system back and forth by pointing at each other. The system successfully moved from person to person over 10 consecutive times.

Tracking an agent While the vision system is recognizing gestures it tracks the person's head. The position of the person's head is converted to a goal for the robot and the robot moves, under local obstacle avoidance towards that goal. The speed of the robot is set at a maximum of 0.4 meters per second, but the robot moves more slowly as it approaches the person it is tracking or as it maneuvers to avoid obstacles. We have successfully tracked people for periods of twenty to thirty minutes in previous work (Huber & Kortenkamp 1995). For this work we added gesture recognition and allowed the person to stop the robot by giving the "halting" gesture. When the robot detects this gesture it stops moving, but the robot's head continues to track the person and the vision system continues to perform gesture recognition. The robot resumes moving when the person gives a "raised" gesture.

Determining tracking height The coarse 3-D model used for gesture recognition requires a rough estimate of the height of the person. For this reason we have implemented a skill that will automatically acquire the height of a person being tracked and reset the 3-D model on-the-run. This skill uses height of the centroid of the head proximity space as the height of the person. Experiments on five people ranging from 1.60m to 1.90m tall showed that the system estimated their height to within an average error of 0.07m. This is well within the gesture recognition system's tolerance for tracking based on a fixed model.

Interpreting gestures in task contexts

Our target environments involve robots working with astronauts in space or on planetary surfaces. Recently, in support of these environments, we have begun to investigate human-robot interaction through gesturing. Wanting to exploit the skills described above in as many situations as possible, we have observed that in many tasks a human pointing gesture can have a wide range of interpretations depending on the task. The middle layer of our 3T architecture is the RAPs system (Firby 1994). A reactive action package (RAP) specifies how and when to carry out routine procedures through conditional sequencing. As such, a RAP provides a way to interpret gestures through context limiting procedures of action.

Finding an agent to track One example of interpreting the same gesture in two different contexts can be shown in the task of pointing out an agent to be tracked. In our research we have noted that the designator agent can simply point to the designated agent's feet and the robot can use the `move-to-point` skill. But when the designated agent is some distance away from the designator, the `acquire-along-vector` skill, while slower, is less error prone. We devised a two step gesture approach wherein the first gesture tells the robot the method to be used to designate the agent to be tracked, and the second gesture would be the pointing gesture itself. Figure 7 shows this RAP (simplified for the purposes of this paper).

```
(define-rap (respond-to-gesture ?agent)
  (method motion-gesture
    (context (or (current-gesture "Pointing")
                 (current-gesture "Halting")))
    (task-net
      (t1 (interpret-gesture-for-motion ?agent))))
  (method normal-acquisition
    (context (current-gesture "Raised"))
    (task-net
      (sequence
        (t1 (speak "Point to the agent's feet"))
        (t2 (interpret-gesture-for-tracking ?agent)))))
  (method long-range-acquisition
    (context (current-gesture "Arched"))
    (task-net
      (sequence
        (t1 (speak "Point at the agent"))
        (t2 (find-agent-along-vector ?agent))))))
```

Figure 7: RAP that uses task context to interpret a gesture.

This RAP assumes a gesture has been received. If it is a pointing or halting gesture, a lower level RAP is called to stop the robot or to move to a point on the floor. If the gesture received is "raised", the usual tracking RAP will be invoked (*interpret-gesture-for-tracking*) which gets a pointing gesture, computes the point on the floor, and then looks for an agent at the appropriate height above that point. If, on the other hand, the arched gesture is detected, the *find-agent-along-vector* RAP will be invoked to get a pointing gesture and find an agent somewhere along the indicated vector. That RAP also enables the tracking skill.

The higher level RAP in Figure 8 sets up a single gesture (such as go to place) or the first of a two gesture sequence. This RAP has three methods depending on whether there is a gesture stored in the RAP memory. Normally, there is no current gesture and the robot must look for the designating agent, get a gesture, and respond appropriately (as described in the previous RAP). Once a gesture task is completed, memory rules associated with lower level RAPs will remove the

```
(define-rap (get-and-respond-to-gesture ?agent)
  (succeed (or (last-result timeout) (last-result succeed)))
  (method no-current-gesture
    (context (not (current-gesture ?g)))
    (task-net
      (sequence
        (t1 (find-agent ?agent))
        (t2 (get-gesture ?agent))
        (t3 (respond-to-gesture ?agent)))))
  (method useful-current-gesture
    (context (and (current-gesture ?g) (not (= ?g "Halting"))))
    (task-net
      (sequence
        (t1 (recognize-gesture ?agent))
        (t2 (respond-to-gesture ?agent)))))
  (method current-halt-gesture
    (context (and (current-gesture ?g) (= ?g "Halting")))
    (task-net
      (sequence
        (t1 (speak "I need another gesture"))
        (t2 (find-agent-at ?agent))
        (t3 (get-gesture ?agent))
        (t4 (respond-to-gesture ?agent))))))
```

Figure 8: RAP that sets up the gesture recognition process.

used gestures from the RAP memory.

But sometimes a lower level RAP will fail, e.g., when the designated agent can't be found, and a gesture such as "Raised" will remain in the RAP memory. Thus, in the second method, an other than halting gesture is current and the robot will turn on the `recognize-gesture` skill (for subsequent gestures) and attempt to carry out (retry) the task indicated by the current gesture.

In some cases, the robot will receive an emergency halting gesture before a lower level RAP is completed such as in the middle of a movement. If this happens the robot's last recollection of a gesture will be "Halting." In these cases, the robot tells the designating agent that they need to start over, and continues as in the first method. These RAPs do not show the details of enabling actual skills, see (Bonasso *et al.* 1995) for details of how this works.

Conclusions

Our goal is to develop technologies that allow for effective human/robot teams in dynamic environments. The ability to use the human's natural communication tendencies allows the robot to be more effective and safer when working among humans. The contributions of our system include a demonstration of gesture recognition in real-time while on-board a mobile robot. The system does not require the user to wear any special equipment nor does it require that the robot, user or background be static. Our contributions also include integrating the gesture recognition system with an intelligent agent architecture that can interpret complex gestures within tasks contexts. This complete system is a first step towards realizing effective human/robot teams. In the future we hope to extend the system by recognizing gestures over time and by integrating gesture recognition with speech recognition.

References

Bonasso, R. P.; Kortenkamp, D.; Miller, D. P.; and Slack, M. 1995. Experiences with an architecture for intelligent, reactive agents. In *Proceedings 1995 IJCAI Workshop on Agent Theories, Architectures, and Languages.*

Borenstein, J., and Koren, Y. 1991. The Vector Field Histogram for fast obstacle-avoidance for mobile robots. *IEEE Journal of Robotics and Automation* 7(3).

Brooks, R. A. 1986. A Robust Layered Control System for a Mobile Robot. *IEEE Journal of Robotics and Automation* 2(1).

Cassell, J. 1995. Speech, action and gestures as context for on-going task-oriented talk. In *Working Notes of the 1995 AAAI Fall Symposium on Embodied Language and Action.*

Darrell, T. J.; Maes, P.; Blumberg, B.; and Pentland, A. 1994. A novel environment for situated vision and behavior. In *Workshop on Visual Behaviors: Computer Vision and Pattern Recognition.*

Firby, R. J. 1994. Task networks for controlling continuous processes. In *Proceedings of the Second International Conference on AI Planning Systems.*

Gavrila, D. M., and Davis, L. 1995. 3-D model-based tracking of human upper body movement: A multi-view approach. In *IEEE Symposium on Computer Vision.*

Huber, E., and Kortenkamp, D. 1995. Using stereo vision to pursue moving agents with a mobile robot. In *1995 IEEE International Conference on Robotics and Automation.*

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.*

Nishihara, H. 1984. Practical real-time imaging stereo matcher. *Optical Engineering* 23(5).

Wilson, A., and Bobick, A. 1995. Configuration states for the representation and recognition of gesture. In *International Workshop on Automatic Face and Gesture Recognition.*