Recognizing and Interpreting Gestures within the Context of an Intelligent Robot Control Architecture

R. Peter Bonasso, Eric Huber and David Kortenkamp
Metrica Inc. Robotics and Automation Group
NASA Johnson Space Center — ER4
Houston, TX 77058
bonasso@aio.jsc.nasa.gov

Abstract

In this paper we describe a stereo vision system that can recognize natural gestures and an intelligent robot control architecture that can interpret these gestures in the context of a task. The intelligent control architecture consists of three layers. The top layer deliberates via a state-based, hierarchical, non-linear planner. The bottom layer consists of a suite of reactive skills that can be configured into synchronous state machines by the middle layer. The middle layer, implemented in the RAPs system, serves to mediate between the long range deliberation of the top layer and the continuous activity of the bottom layer. It is in the middle layer that we are investigating context focused deictic gesturing for human-robot interaction. When directed in the context of different RAPs, human gestures can be interpreted by the system differently for different tasks. This work shows that a) our architecture, designed to support perception and action, can also support other forms of communication, and b) task contexts can act as resources for communication by simplifying the interpretation of communicative acts.

Introduction

We are in the process of developing a mobile robot system that can interpret gestures within the context of a task. The robot uses a stereo vision system (described in the next section) to recognize natural gestures such as pointing and hand signals and then interprets these gestures within the context of an intelligent agent architecture (described in the next subsection). While the system is not yet fully implemented, we feel we have enough of the pieces to describe how it would apply to the fetching and delivering task described in the symposium call for papers.

Overview of the architecture

Since the late eighties we have investigated ways to combine deliberation and reactivity in robot control architectures (Sanborn, Bloom, & McGrath 1989; Bonasso 1991; Bonasso *et al.* 1995) in order to program robots to carry out tasks robustly in field environments. The robot control software architecture we are using is an outgrowth of several lines of situated reasoning research in robot intelligence (Firby 1989; Gat 1992; Connell 1992; Slack 1992; Yu, Slack, & Miller 1994; Elsaesser & Slack 1994) and has proven useful for enabling mobile robots to accomplish tasks in field environments. This architecture separates the general robot intelligence problem into three interacting pieces:

- A set of robotic specific reactive skills. For example, grasping, object tracking, and local navigation. These are tightly bound to the specific hardware of the robot and must interact with the world in real-time.
- A sequencing capability which can differentially activate the reactive skills in order to direct changes in the state of the world and accomplish specific tasks. For example, exiting a room might be orchestrated through the use of reactive skills for door tracking, local navigation, grasping, and pulling. In each of these phases of operation, the skills of the lower-level are connected to function as what might be called a "Brooksian" robot (Brooks 1986) – a collection of networked state machines. We are using the Reactive Action Packages (RAPs) system (Firby 1989) for this portion of the architecture.
- A deliberative planning capability which reasons in depth about goals, resources and timing constraints. We are using a state-based non-linear hierarchical planner known as AP (Elsaesser & MacMillan 1991), the adversarial planner, since it grew out of research in counterplanning. AP is a multi-agent planner which can reason about metric time for scheduling, monitor the execution of its plans, and replan accordingly.

These capabilities allow a robot, for example, to accept guidance from a human supervisor, plan a series

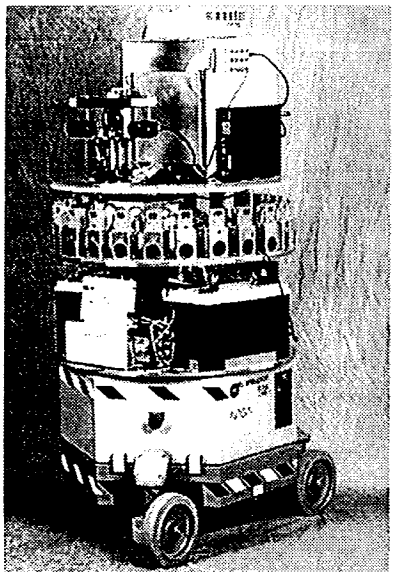

Figure 1: A mobile robot with an on-board vision system.

of activities at various locations, move among the locations carrying out the activities, and simultaneously avoid danger and maintain nominal resource levels. The system has been shown to provide a high level of human supervision that preserves safety while allowing for task level direction, reaction to out-of-norm parameters, and human interaction at all levels of control.

Overview of visual skills

We currently have our stereo system mounted on a table. In this configuration we have tested various gesture recognition algorithms, as described in Section 2. This system will soon be mounted on a mobile robot (see Figure 1). The vision system will provide the following skills:

- DETECT-VECTOR: Determines the arm that is pointing and the three-dimensional vector that defines the pointing direction.
- TRACK-VECTOR-TO-POINT: Moves the robot's binocular head along the pointing vector until it encounters a distinct object or a timeout has been reached.
- DETECT-GESTURE: Determines whether the right, the left or both arms have been raised above the person's head.
- DETECT-HALT-GESTURE: Determines whether one or both of the person's arms are directed straight at the robot.
- TRACK-AGENT: Moves the robot's binocular head to keep the person in the center of the field of view.
- LOCATE-OBJECT-OF-SIZE: Locates an object of a particular, approximate size in the field of view.
- SPEAK: generates a voice statement (from the robot, not the cameras).

In addition, each skill has an event skill associated with it which detects the completion of the skill and returns any necessary variables. One such event skill for the DETECT-VECTOR SKILL above would be VECTOR-DETECTED, which would return the location and direction of the vector.

In the next section of this paper, we briefly describe how our stereo vision system recognizes the gesturing skills. Following that we describe how these skills are used in task context.

Visual skills

The stereo vision system we use for gesture recognition consists of two B&W CCD cameras mounted on a pan-tilt-verge head. In order to efficiently process the enormous amount of information available from the cameras, we use techniques that have recently been developed by the active vision research community (Ballard 1991; Coombs & Brown 1991). In particular, we address the issue of *gaze control*, i.e., where to focus attention and visual resources. The basis of our stereo vision system is the PRISM-3 system developed by Keith Nishihara (Nishihara 1984). First, we give a short overview of the PRISM-3 vision system, then we describe how we extended the PRISM-3 system for gesture recognition.

The PRISM vision system

The PRISM-3 stereo vision system is the embodiment of Keith Nishihara's sign correlation theory (Nishihara 1984). This theory is an extension of work by Marr and Poggio (Marr & Poggio 1979) that resulted in a stereo matching algorithm consistent with psychophysical data. The PRISM system employs dedicated hardware including a Laplacian-of-Gaussian (LOG) convolver and a parallel sign correlator to compute spatial and/or temporal disparities between correlation patches in each image (see Figure 2).

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

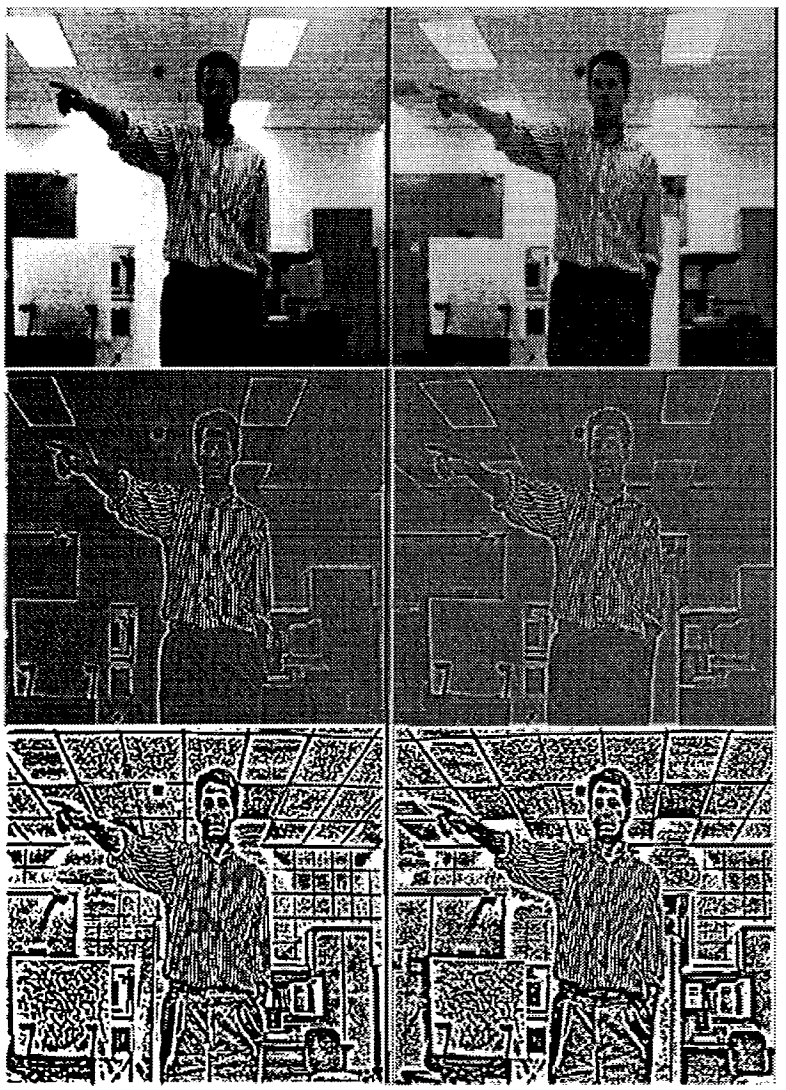

Figure 2: Top: Stereo intensity pair; Middle: Stereo Laplacian of Guassian filter pair; Bottom: Stereo sign of Laplacian of Gaussian pair

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., LOG filter variations in light intensity across a region of the image.

In order to achieve visual tasks in our system, we have several behaviors that assess information within the proximity space in order to effect the future position of 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. We have the following behaviors that operate on our proximity space:

- **Follow**: This behavior takes an average of several correlation based motion measurements within a proximity space in order to produce a 2-d force vector (virtual) in the direction of motion.
- **Cling**: This behavior is attracted to surfaces and produces a vector that tends to make the proximity space "cling" to them.
- **Migrate**: The migration behavior influences the proximity space to traverse a surface until it reaches the surface boundary where it eventually runs out of material and "stalls.".
- **Pull**: This very simple but useful behavior produces a pull vector from the centroid of the proximity space toward the stereo head. This vector tends to move the proximity space toward local depth minima.
- **Resize**: This behavior influences the size of the proximity space inversely proportionally to the degree of occupancy, also changing the density of measurements within the space. The net effect of this behavior is that it tends to increase the search space if an object is "getting away" or even momentarily lost.

We have had much success in building various forms of complex tracking behaviors from sets of the behaviors described above (Huber & Kortenkamp 1995). The Follow behavior provides the foundation for tracking fast moving objects. It provides the means for quick response, but as explained, fails to maintain track. However, the Cling behavior picks up where the Follow behavior fails. It shuffles the proximity space back onto the object's surface whenever it nears a boundary.

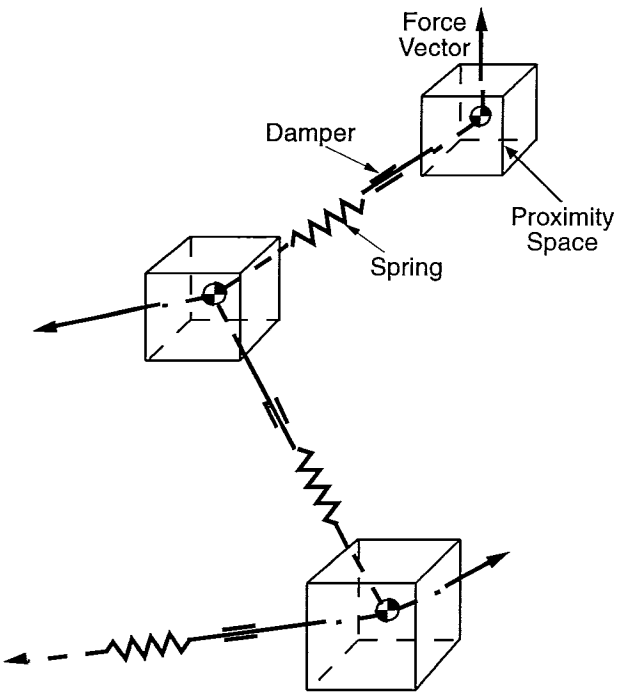

Figure 3: Proximity spaces can be chained together with kinetic links.

Chaining proximity spaces

While a single proximity space is sufficient for tracking a single moving object, multiple proximity spaces must be used in order to recognize gestures. The multiple proximity spaces track specific human joints as they move. We have had particular success when using chains of proximity spaces. Chaining places relative constraints on the position of proximity spaces thus bounding their travel. Although the action of a proximity chain is dynamic, it is actually derived from a simple kinematic model. The kinematic model is composed of spring/damper links which are used to connect proximity spaces (see Figure 3). The proximity spaces are co-located with the human elbow and shoulder joints. This model includes a relaxed or default position for each joint. Through appropriate selection of joint angles the system can be biased to "expect" certain configurations of the object being tracked. For the purpose of human gesture recognition, the unsprung proximity spaces span laterally from the upper torso.

Figure 4: Image showing the location of proximity spaces that are used to find the pointing vector.

Figure 5: Image showing the location of proximity spaces that are used for gesture recognition.

There, they are ready to "cling onto" the arms as soon as they are extended in a gesturing configuration. As the proximity spaces attach to the arm their individual migration vectors pull them down the arm causing them to span out as the arm is detected and collapse into a more compact configuration when not needed. Currently, this is limited to orientations in which the human is roughly facing towards or away from the stereo head.

The head is used as the foundation for gesture recognition. From the head we branch proximity spaces for both shoulders, elbows and wrists. All of the proximity spaces are connected by kinematic chains. Figure 4 shows the head proximity space and three proximity spaces for each arm. The subject's left arm is relaxed and the proximity space simply collapse to the shoulder and "wait" for the arm to be extended. The subject's right arm is extended and the proximity spaces follow it out. The kinematic links between proximity spaces serve to limit their travel. To determine a gesture direction, one simply needs to know the three dimensional location of the shoulder proximity space and the hand proximity space. In Figure 5, using the same proximity spaces we can recognize a gesture defined by the subject's right hand being above their head. The robot control architecture can assign a particular interpretation to this gesture.

Interpreting gestures in context

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. Our RAPs system provides a way to disambiguate such gestures through context limiting procedures of action.

Fetching an object

A RAP is a structure that shows how and when to carry out procedures through conditional sequencing. While we have not yet integrated the above described set of primitive vision skills to RAPs, we envision the following kind of RAP that would use some of these skills to fetch an object:

```
(define-rap (fetch-object ?color)
  (succeed (and (location robot ?finalx
                          ?finaly ?finalz)
                (holding-object-at robot
                                   ?x ?y ?z)))
  (method size-locate-object
   (context (size-detection-available))
    (sequence
     (t1 (speak "Please point to the ~a object."
                ?color))
     (t2 (detect-vector)
```

```
      (wait-for (vector-detected ?vx ?vy ?vz
                                 ?roll ?pitch
                                 ?yaw)
          :succeed))
    (t3 (locate-object-of-size ?size ?vx ?vy ?vz
                               ?roll ?pitch ?yaw)
      (wait-for (object-of-size-found ?x ?y ?z)
                :succeed)
      (wait-for (object-not-found)
                :succeed (object-not-found
                          ?size)))
    (t4 (robot-move ?x ?y ?z))
    (t5 (grasp-object-at ?x ?y ?z))))
  (method cannot-locate
   (context (or (not (size-detection-available))
                (object-not-found ?size)))
    (sequence
     (t1 (speak "Please point to the object
                 of size ~a" ?size))
     (t2 (detect-vector)
      (wait-for (vector-detected ?vx ?vy ?vz
                                 ?roll ?pitch
                                 ?yaw)
          :succeed))
     (t3 (track-vector-to-point ?vx ?vy ?vz ?roll
                                ?pitch ?yaw)
      (wait-for (track-done ?x ?y ?z)
                :succeed))
     (t4 (robot-move ?x ?y ?z))
     (t5 (grasp-object-at ?x ?y ?z)))))
```

There are memory rules used to post information that is accessible to any RAP. For this discussion assume there is a memory rule associated with the grasp-object-at RAP which will assert (holding-object-at robot ?x ?y ?z), a memory-rule associated with the robot-move RAP which will assert (location robot ?finalx ?finaly ?finalz), and finally, a memory rule associated with the above RAP as follows:

```
(define-memory-rule (fetch-object ?size) :event
  (match-result ((object-not-found ?size)
   (rule (t
         (mem-add (object-not-found ?size)))))))
```

The RAP interpreter first checks the succeed clause and finding it not to be true, selects a method. If the robot has models of object sizes available, the first method is selected. In this method, through a pointing gesture, the user provides the direction for the vision system to start its search for the object. If the object is not found, the variables ?x ?y and ?z are not bound and the method fails. The memory rule fires on the not found event. The RAP interpreter then tries the next method. This method would also be the first method used if the robot had no size models available. In the second method, the same human gesture actually indicates the object to be grasped, rather than just the general direction. Assuming the robot-move and grasp-object-at RAPs are successful, the proper assertions will be made in memory to allow the succeed clause to be true.

The above example shows the power of the bottom two tiers of our architecture to allow for deictic gestures to be interpreted in the same framework as the rest of the robot's sensing and acting.

The human gesture is directly interpreted in the context of the current RAP method, i.e., the variable bindings of the method focus the interpretation of the gesture. In the first method, the pointing gesture is interpreted as a direction for search. The same gesture in the second method, combined with an additional skill (track-vector-to-point), indicates an object rather than a direction. The semantics of the gesture are unambiguous at the time of the act.

The above example can also serve to illustrate an answer to the question of how can agents mediate between the propositional representations of language and the non-propositional representations of perception and action. Assume that the instruction to the robot, accepted at the planning level (where a discourse processor would interface with the planner) was "Take a large object to the work bench". The resulting plan would bottom out in several primitive operations, each representing a RAP. One of these would be fetch-object which would be completed when the succeed clause of the above RAP was established in the RAP memory. At this point the planner would generate a token and assert among other things (and (object sym37) (color sym37 red) (location sym37 (?x ?y ?z))) in its own memory. The RAPs links the location of the object as detected by the vision system with the token sym37 in the planner's memory. In this way, RAPs serves as a kind of syntactic differential between the propositional representation of the planner and the indicators coming from the vision system.

Delivering an object

We are investigating a number of different uses for the PRISM vision skills via our architecture. Imagine, after asking the robot to fetch an object, you now want it delivered to Major Kira (the robot has (size Kira short) in its RAP memory). The resulting RAP might be :

```
(define-rap (deliver-to-agent ?agent)
  (succeed (and (in-view ?agent)
                (halt-signaled-from ?agent)))
    (method agent-in-view
      (context (in-view ?agent))
      (parallel
        (t1 (track-agent ?agent))
        (t2 (detect-halt-gesture ?agent)
            (wait-for (halt-gesture-detected
                       ?agent) :succeed))))
    (method agent-not-in-view
```

```
(context (and (not (in-view ?agent))
              (size ?agent ?size)
         (in-view ?someone)))
(sequence
  (t1 (speak ''Where is ~a?'' ?agent))
  (t2 (get-location-from-agent ?someone =>
                               ?x ?y ?z))
  (t3 (find-object-of-size-near ?size ?x
                                ?y ?z =>
                                ?object))
  (parallel
    (t4 (track-agent ?object))
    (t5 (detect-halt-gesture ?object)
        (wait-for (halt-gesture-detected
                   ?object) :succeed))))))
```

Get-location-from-agent and find-object-of-size-near are themselves RAPs which use the detect-vector, track-vector, and locate-object-of-size skills and attendant events. The => symbol makes bound variables available in the task-net. The associated memory rule would be:

```
(define-memory-rule (follow-agent ?agent) :event
  (match-result
    (succeed
       (rule (t
              (mem-add (halt-signaled-from
                        ?agent)))))))
```

Here, Kira, the assumed binding for ?agent, is distinguished by size and a location indicated by a pointing gesture. She may have already been in view when the command is given, but if not, the robot asks you where to look for her. In either case, the robot follows Kira (or the short object first found at location ?x ?y ?z) until she signals for the robot to stop.

Future work and conclusions

We have demonstrated a stereo vision system that can recognize natural gestures and an intelligent control architecture that can interpret these gestures in the context of a task. While the two have not yet been brought together, we anticipate doing so in the near future. We also have a second robot that can detect color. We are experimenting with ways of having RAPs coordinate these two robots to accomplish tasks that require both gesturing and color detection. The robot with color vision also has been programmed to recognize human faces (Wong, Kortenkamp, & Speich 1995). We also are planning to add a simple speech recognition capability to both robots. With gesturing, face recognition, and speech recognition we will have a rich environment in which to explore human-robot interfacing capabilities.

References

Ballard, D. H. 1991. Animate vision. *Artificial Intelligence* 49(1).

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.*

Bonasso, R. P. 1991. Integrating reaction plans and layered competences through synchronous control. In *Proceedings International Joint Conferences on Artificial Intelligence.*

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

Connell, J. H. 1992. SSS: A hybrid architecture applied to robot navigation. In *Proceedings IEEE International Conference on Robotics and Automation.*

Coombs, D. J., and Brown, C. M. 1991. Cooperative gaze holding in binocular vision. In *Proceedings of the Fifth IEEE International Symposium on Intelligent Control.*

Elsaesser, C., and MacMillan, R. 1991. Representation and algorithms for multiagent adversarial planning. Technical Report MTR-91W000207, The MITRE Corporation.

Elsaesser, C., and Slack, M. G. 1994. Integrating deliberative planning in a robot architecture. In *Proceedings of the AIAA/NASA Conference on Intelligent Robots in Field, Factory, Service, and Space (CIRFFSS '94).*

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

Gat, E. 1992. Integrating planning and reacting in a heterogeneous asynchronous architecture for controlling real-world mobile robots. In *Proceedings of the National Conference on Artificial Intelligence (AAAI).*

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.*

Marr, D., and Poggio, T. 1979. A computational theory of human stereo vision. In *Proceedings of the Royal Society of London.*

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

Sanborn, J.; Bloom, B.; and McGrath, D. 1989. A situated reasoning architecture for space-based repair and replace tasks. In *Goddard Conference on Space*

Applications of Artificial Intelligence (NASA Publication 3033).

Slack, M. G. 1992. Sequencing formally defined reactions for robotic activity: Integrating raps and gapps. In *Proceedings of SPIE's Conference on Sensor Fusion.*

Wong, C.; Kortenkamp, D.; and Speich, M. 1995. A mobile robot that recognizes people. To appear in *Proceedings of the 1995 IEEE International Conference on Tools with Artificial Intelligence.*

Yu, S. T.; Slack, M. G.; and Miller, D. P. 1994. A streamlined software environment for situated skills. In *Proceedings of the AIAA/NASA Conference on Intelligent Robots in Field, Factory, Service, and Space (CIRFFSS '94).*