AI Magazine Volume 24 Number 2 (2003) (© AAAI)

GRACE
An Autonomous Robot for the AAAI Robot Challenge

Reid Simmons, Dani Goldberg, Adam Goode, Michael Montemerlo, Nicholas Roy, Brennan Sellner, Chris Urmson, Alan Schultz, Myriam Abramson, William Adams, Amin Atrash, Magda Bugajska, Michael Coblenz, Matt MacMahon, Dennis Perzanowski, Ian Horswill, Robert Zubek, David Kortenkamp, Bryn Wolfe, Tod Milam, and Bruce Maxwell

■ In an attempt to solve as much of the AAAI Robot Challenge as possible, five research institutions representing academia, industry, and government integrated their research into a single robot named GRACE. This article describes this first-year effort by the GRACE team, including not only the various techniques each participant brought to GRACE but also the difficult integration effort itself.

The AAAI Robot Challenge was established four years ago as a grand challenge for mobile robots. The main objectives of the challenge are to (1) provide a task that will demonstrate a high level of intelligence and autonomy for robots acting in a natural, peopled, dynamic environment; (2) stimulate state-of-the-art robotics research to address this task; and (3) use robot demonstrations to educate the public about the exciting and difficult challenges of robotics research. The challenge was designed as a problem that would probably need a decade to achieve adequately. When the challenge was designed, it was anticipated that no single research institution would have adequate resources to meet the challenge on its own.[1]

The challenge task is for a robot to participate in the American Association for Artificial Intelligence National Conference on Artificial Intelligence—the robot must find the registration booth and register, interacting with people as needed, then with a map in hand, find its way to a location in time to give a technical talk about itself.[2] Ideally, the robot should be given no more information than any other participant arriving in a new city to attend a major technical conference. In particular, the robot should not know the layout of the convention center beforehand, and the environment should not be modified. Practically, however, the organizers understand that compromises and flexibility will be necessary to get current state-of-the-art robots to achieve the task.

There are a number of important technologies that are needed to meet the challenge. These technologies include localization in a dynamic environment; safe navigation in the presence of moving people; path planning; dynamic replanning; visual tracking of people, signs, and landmarks; gesture and face recognition; speech recognition and natural language understanding; speech generation; knowledge representation; and social interaction with people. Although researchers have worked on all these areas, they all need further work to be robust in the environment that the challenge specifies. In addition, a major challenge is the integration of these technologies.

In August 2001, several of the authors agreed to join efforts to attempt the challenge in its entirety. We had each been working on technologies related to the challenge and felt that by pooling our efforts we could do reasonably well. In addition, we believed that the type of collaborative work that was needed to pull this off would help advance robotics. We realized that integrating hardware and software from five institutions would be difficult. Our first-year goal, therefore, was to create architecture and infrastructure to integrate our existing software into a system that could do a credible job with the challenge task. We all agreed that this would be a multiyear effort

Figure 1. The Robot GRACE.

and that in subsequent years, we would build on this year's robot system.

In e-mail conversations and meetings during the winter of 2002, we formulated the basic approach and architecture. We decided that there were several possible approaches: (1) we could bring our own robots and each do part of the task, "handing off" from one to another, (2) we could use a common hardware platform but use our own existing software, or (3) we could do a full-blown hardware and software integration. We quickly agreed to try for option 3, but option 2 would be a good fallback position. We spent the spring of 2002 converting existing software to run on the common hardware platform (see Robot Hardware) and common integration architecture (see Software Architecture). In the end, we achieved some- where between options 2 and 3, with the robot successfully performing most of the major subtasks with little human intervention (see Doing the Challenge Task). In July 2002, we traveled to the Eighteenth National Conference on Artificial Intelligence at the Shaw Convention Centre in Edmonton, Alberta, to take part in the challenge.

Robot Hardware

GRACE (graduate robot attending conference) is built on top of a B21 mobile robot built by RWI. GRACE has an expressive computer-animated face projected on a 15" flat-panel LCD screen as well as a large array of sensors (figure 1). The sensors that come standard with the B21 include touch, infrared, and sonar. Near the base is a SICK scanning laser range finder that provides a 180-degree field of view.

At one of our first meetings, we discussed the various hardware each team would need to integrate into the Carnegie Mellon University (CMU) platform. GRACE has several cameras, including a stereo camera head on a pan-tilt unit built by Metrica TRACLabs and a single color camera with pan-tilt-zoom capability built by Canon. GRACE can speak using a high-quality speech-generation software (FESTIVAL) and receive speech responses using a wireless microphone headset (a Shure TC computer wireless transmitter-receiver pair).

GRACE runs all software on board. Two 500-megahertz processors, running LINUX, run most of the autonomy software. A Sony VAIO PICTUREBOOK laptop, running WINDOWS, runs the speech-recognition software. In addition, there is a separate processor for the Metrica stereo head and a Linksys wireless access point to connect the robot to the outside world (for debugging, monitoring, and controlling the presentation during the talk).

Software Architecture

One of the more difficult parts of the challenge for us was determining how to integrate a vast amount of software that had been developed by the participating institutions, mostly on different hardware platforms.[3] Early on, we decided to integrate everything onto a common hardware platform, as described previously, with different groups providing software services that would interface to various pieces of hardware. The idea was that the services would abstract away details of the actual hardware

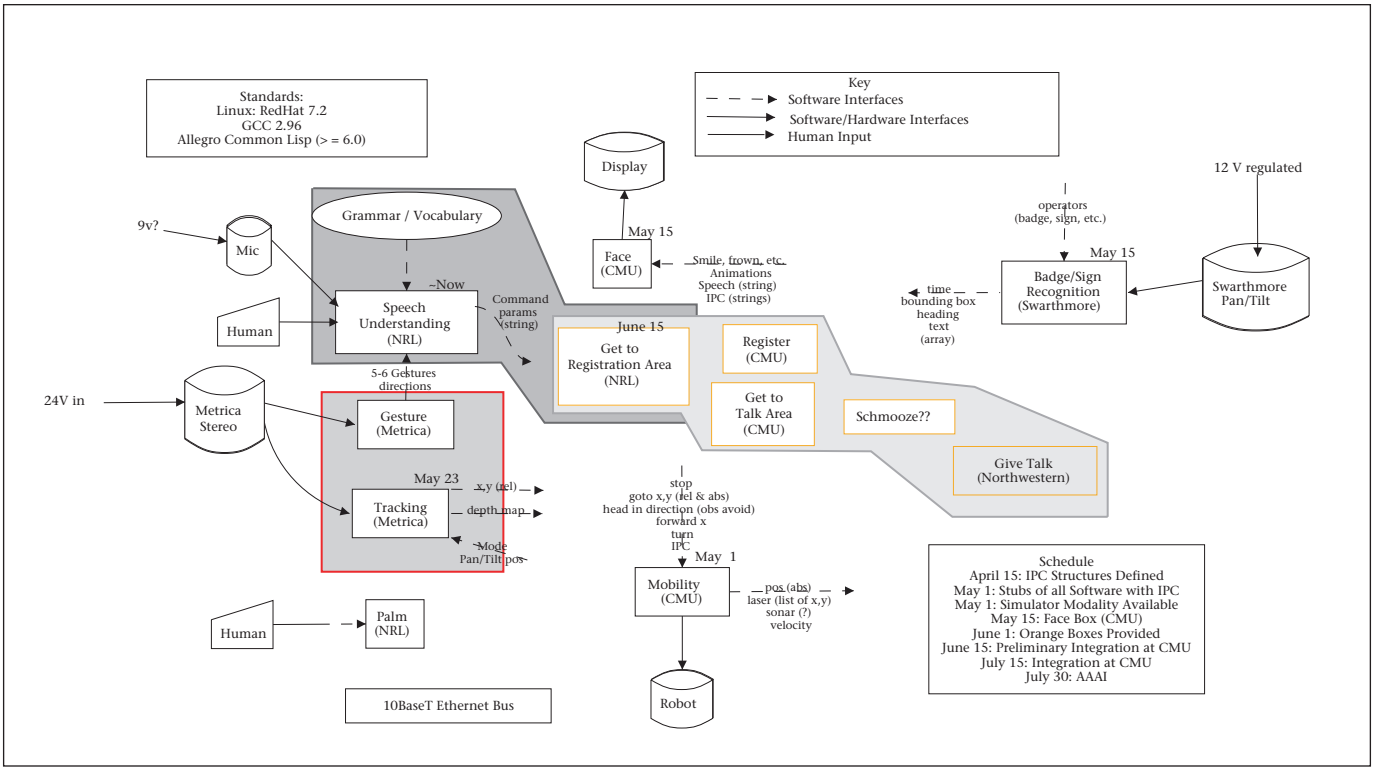


Figure 2. GRACE *Software Architecture Diagram.*

platform, making subsequent development easier. Development of interfaces between modules occupied the bulk of our initial work. Each team needed to define the input and output of their modules and work out details of how the various modules would interact. In particular, CMU provided interfaces to the robot base (motion and localization), speech generation, and computer-animated face; the Naval Research Laboratory (NRL) provided speech recognition and natural language–understanding interfaces; Swarthmore College provided vision-processing code and control over the Canon pan-tilt-zoom camera; and Metrica provided stereo vision and control over their pan-tilt head. In addition, CMU provided a simple graphic simulator so that programs could be tested remotely, in advance of integration on the actual robot platform.

Software for the various subtasks was then built on top of these services. Although the services, for the most part, were task independent, the software that ran the various tasks was a mixture of task-independent and task-dependent code. In particular, NRL was responsible for the part of the challenge when the robot entered the conference center to when it neared the registration booth; CMU was responsible for elevator riding, getting in line at the registration booth (using Swarthmore's vision system), registering for the conference, and navigating to the lecture area; and Northwestern University was responsible for having GRACE give its talk. Figure 2 presents a high-level view of the software architecture and development responsibilities. The section entitled Doing the Challenge Task presents details of the task-level software.

To facilitate distributed development and simplify testing and debugging, the GRACE system was designed as a set of independent programs that communicated using message passing. The interprocess communication (IPC) package was chosen for (nearly all) communications because of its expressiveness, ease of use, and familiarity by some of the teams (both CMU and Metrica have used IPC in the past).[4] As much as possible, all software was to be written in C or C++ (using the GCC 2.96 compiler), running under RED HAT LINUX 7.2. Exceptions included the use of a WINDOWS laptop to run VIAVOICE,[5] the use of Allegro Common Lisp for NRL's NAUTILUS natural language–understanding system, and the use of SWIG and PYTHON for the elevator riding code. In addition, OPENGL, PERL, and FESTIVAL were used for the computer-animated face and speech generation.[6]

Finally, the computer-animated face and sev-

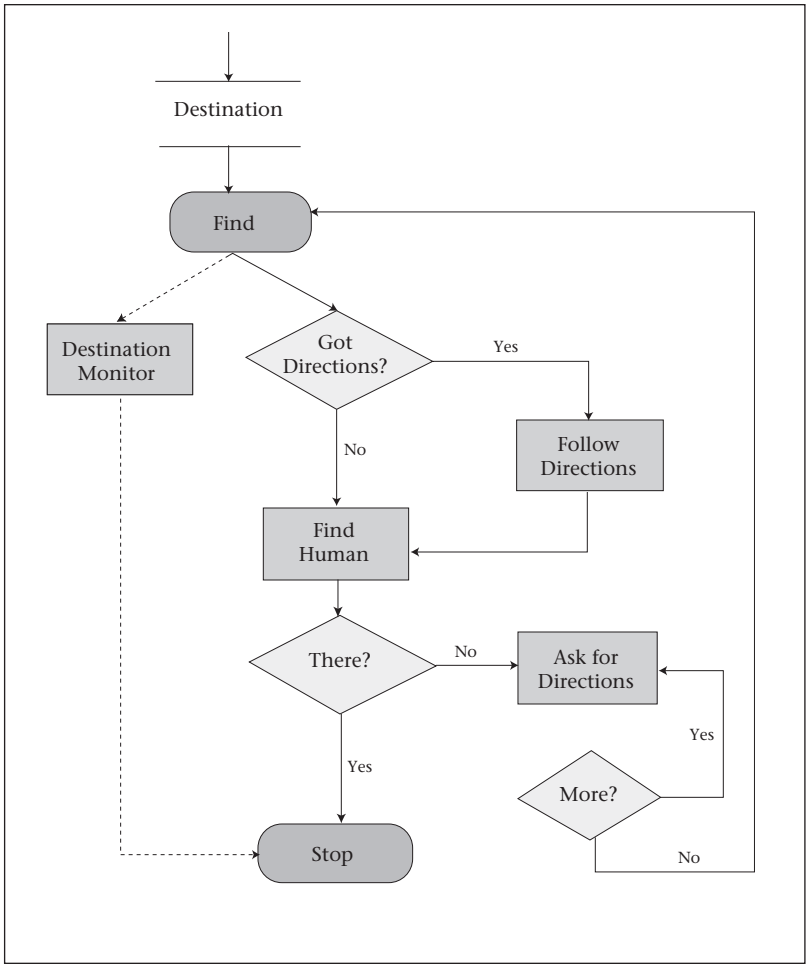


Figure 3. Direction Taking.

eral of the task-level programs were written using the TASK DESCRIPTION LANGUAGE (TDL). TDL is an extension of C++ that contains explicit syntax to support hierarchical task decomposition, task synchronization, execution monitoring, and exception handling (Simmons and Apfelbaum 1998).[7] A compiler translates TDL code into pure C++ code that includes calls to a domain-independent TASK-CONTROL MANAGEMENT (TCM) library. The translated code can then be compiled using standard C++ compilers linked with other software. The idea is to enable complex task-level control constructs to be described easily, enabling developers to focus more on the domain-dependent aspects of their programs.

Doing the Challenge Task

As mentioned previously, the challenge is to have an autonomous mobile robot attend the National Conference on Artificial Intelligence. More specifically, the challenge rules are to have the robot perform the following subtasks:[8]

First, starting at the front door of the conference center, navigate to the registration desk (ideally by locating signs, asking people, or following people—at this point, the robot does not have a map of the building).

Second, register. Stand in line if necessary; have the robot identify itself; and receive registration material, a map of the conference center, and a room number and time for its talk.

Third, interact with other conference attendees (ideally recognize participants by reading name tags or recognizing faces and schmooze, striking up brief personal conversations).

Fourth, if requested, perform volunteer tasks as time permits, such as "guarding" a room or delivering an object to another room.

Fifth, get to the conference room on time, using map received in step 3. This step might involve riding an escalator or elevator.

Sixth, make a two-minute presentation about its own technology and answer questions.

For our first year at the challenge, we decided to do all the subtasks except steps 3 (schmoozing) and 4 (volunteer duties) and have the robot itself answer questions from the audience. In addition, the human interaction in step 1 was limited to interaction with one person, a student who worked with the team that summer. In future years, we will expand the scope to include all subtasks and enable arbitrary conference participants to interact with the robot.

The next subsections describe in more detail the major subsystems for each of the challenge tasks.

Getting to the Registration Area

GRACE must start at the entrance to the conference center and find the registration area by interacting with people. Remember that GRACE does not have a map until it reaches the registration desk. This part of the challenge is meant to demonstrate robot interaction with people.

We endowed GRACE with the capability to interact with people using both speech and natural gestures, in particular to allow GRACE to ask for, understand, and follow directions. Using TDL (described in Software Architecture), we created a push-down automata that allowed GRACE to maintain multiple goals such as using an elevator to get to a particular floor and following directions to find the elevator.

We used an off-the-shelf speech-recognition system, IBM's VIAVOICE, to convert spoken utterances to text strings. The text strings were then parsed and interpreted using NAUTILUS, NRL's in-house natural language–understanding system (Perzanowski, Schultz, and Adams 1998; Perzanowski et. al. 2002, 2001; Wauchope 1994). The output of this component is a logical form similar to standard predicate log-

ic. This representation is then mapped to a message, or a series of messages, which is then sent to other modules through an IPC interface that was developed specifically for the challenge.

The a priori top-level goal is to find the registration desk. Additional goals are created as GRACE interacts with people to determine the directions to the registration desk and intermediate locations on the way to the registration desk. To achieve a goal, we interleave linguistic and visual information with direction execution (figure 3). If there are no directions to be followed, GRACE performs a random walk until a human is detected (for the challenge in 2002, human detection was done using a laser scanner; in future years, we will incorporate vision-based detection of people). GRACE then engages the human in a conversation to obtain directions to the destination in question. Simple commands, such as "turn left" and "go forward five meters," as well as higher-level instructions, such as "take the elevator" and "turn left next to the elevator," are acceptable (note that in the Shaw Convention Centre, one needed to take an elevator down two flights from the entrance to get to the registration area). In addition, GRACE can ask questions such as "Am I at the registration desk?" and "Is this the elevator?" The task is completed once the destination is reached, as determined by an explicit human confirmation or perception of the goal.

Besides accepting speech input, GRACE can incorporate gestures, such as when a human points to a given location. Initially, we were planning on using stereo-based vision to track both people and their gestures, but this part of the software was not ready in time. As a last-minute backup, we developed a personal digital assistant (PDA)–based interface in which movements of the stylus on the screen were interpreted as directional gestures.

Execution monitors run concurrently to ensure both safety and the integration of various required linguistic and sensory information. For example, an explicit stop command can be issued if unforeseen or dangerous conditions arise. Also, perception processing occurs concurrently with interaction, allowing the detection of the destination or a human to be interleaved with other information required to perform the task.

Two types of direction can be given. For a simple action command, such as "turn left," the action is queued until the speaker is done giving directions, then the actions are executed sequentially. The second type of command is an instruction specifying an intermediate destination, such as "take the elevator to the second floor." In this case, an intermediate goal is instantiated (getting to the elevator), and the logic is recursively applied to the new goal (figure 4). All directions to this point are regarded as directions to the subgoal, and subsequent directions are associated with the parent goal. Once all the available directions have been executed, GRACE concludes that either it has arrived at the destination, or additional information is required to reach the goal. If GRACE perceives the destination before all the directions are executed, the remaining ones are abandoned, and it continues with the next goal.

Figure 4. Human Giving Directions to GRACE *to Find Elevator out of View to the Right.*

Thus, if GRACE asks a human bystander "Excuse me, where is the registration desk?" and the human responds "GRACE, to get to the registration desk, go over there <accompanied by a gesture>, take the elevator to the ground floor, turn right, and go forward 50 meters," the human's input is mapped to a representation similar to the following:

```
Find Registration Desk:
   Find Elevator (ground floor);
      Go over there <gesture>;
   Turn right;
   Go forward 50 meters.
```

Once GRACE has found the elevator, control is temporarily turned over to CMU's elevator riding process (see subsection entitled Riding the Elevator). When GRACE determines that it is within a reasonably close distance to the registration desk, the find-the-desk process is terminated, and control is given to the process that approaches the registration desk (see Finding the Registration Booth).

Figure 5. The Elevator in the Shaw Convention Centre.

Riding the Elevator

As mentioned previously, the registration area in the Shaw Convention Centre was not on the same floor as the street entrance. Our choices in addressing this situation were rather limited—stairs are out of the question, and escalators are no good either. The only viable alternative was to have GRACE ride the elevator (figure 5).

The first problem is to find the elevator itself. We assume that the system has brought the robot near the elevator and pointed it generally to face that direction. Thus, the laser should have a good view of the elevator, and the robot will just need to perceive the unique signature of the elevator doors in the laser readings and get itself lined up with the doors. For example, given that the robot is positioned as shown in figure 6, the system will see laser readings such as those in figure 7.

Although people can readily make out the shapes of the elevators in the laser points, having the robot find the elevators is unfortunately a bit more involved. The algorithm that we developed to perceive elevators from laser scans is as follows:

Straighten out the view of the world

Find horizontal segments corresponding to bits of walls

Filter the segments to eliminate noise and impossible conditions

Merge small, adjacent segments into single segments

Use feature matching to find possible elevators

Filter out impossible elevators

This process is iterative and constantly running. The robot starts by attempting to fit straight lines to points it sees. Using these lines, it comes up with a guess of how far off it is from facing the wall. It then "mentally" rotates the points in the world and tries again. Fairly quickly, the walls slide into place, and the system can detect the characteristic shape of elevator doors.

The system uses a feature-based recognizer to detect elevators. Given the transformation of the input points, it is sufficient to consider only horizontal segments within some parameterized tolerances for length and offset. In general, the system looks for three characteristic shapes. The first shape is a standard elevator inset (figure 8). Because elevators are generally of a certain width, but also have a deeper inset than office doors, the inset information can fairly reliably pick out an elevator from an office or conference room. The second two shapes are similar to the first but with some information removed. Although these shapes are still valid elevator candidates, the robot would probably need to move around a bit to get a better view of the elevator to make a final determination. When these patterns are applied to the input data in figure 7, the system detects the two elevators shown in figure 9.

One difficulty is that some patterns that are

not elevators can actually look similar to the patterns in figure 8. For example, figure 10 illustrates two types of patterns that are not elevators. Note that in practice, some patterns that initially look good (for example, the two patterns on the right in figure 8) might actually turn out to be bad patterns when more information is acquired (by moving around).

After the robot detects an elevator, it gets into position and waits for the door to open. Although the laser can often see several elevators simultaneously, the robot cannot safely move fast enough if a door opens too far away. Thus, the robot picks one elevator to wait in front of and moves only if it later decides that a better elevator pattern is nearby. Specifically, it waits for a while and, after a timeout with no activity, searches and lines itself up again.

Once it has chosen an elevator and moved in front of it, the robot waits for some time for the door to open. If the door opens soon enough (as shown by the laser readings), the robot navigates in and turns around. When it has determined (by human interaction or other means) that it is on the destination floor, it moves out of the elevator when the path is clear.

Although the elevator-riding program worked well in testing, two main problems were encountered when we arrived in Edmonton. First, the area surrounding the elevator, and the elevator itself, was made primarily of laser-invisible glass (figure 5). To solve this problem, we discretely put a single strip of stylish green tape all around the area, just at laser height. This tape neatly solved the problem and drew little attention from the onlookers. The second problem was that the elevator pattern on the entrance floor of the convention center was quite unusual. The elevator had a normal inset on its left but abutted a long wall on its right (figure 11). The solution was to adjust the feature-based recognizer to accept this pattern as a valid elevator. Clearly, though, this type of tweaking is not a general solution to the problem.

With these problems solved, the elevator-riding portion of the challenge went quite well. However, there are a few issues still remaining. The most visible issue relates to the slowness of the error-correcting actions. For example, when the robot is misaligned in the elevator, it waits for a long time before it decides to back up and try again. It should detect and recover from these kinds of errors much faster. Second, as pointed out earlier, a more general recognizer needs to be developed—perhaps one that uses both laser and vision. Finally, the robot needs to be able to detect for itself when it is on the correct floor. We are currently developing a

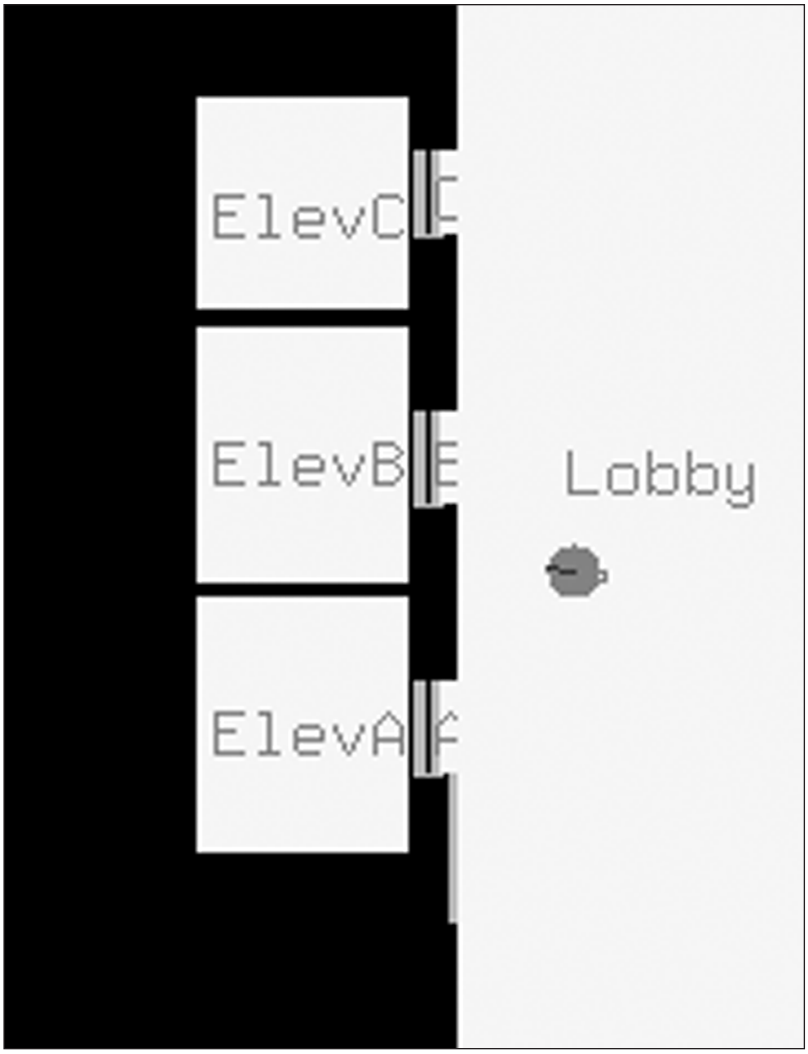


Figure 6. The Simulation Environment.

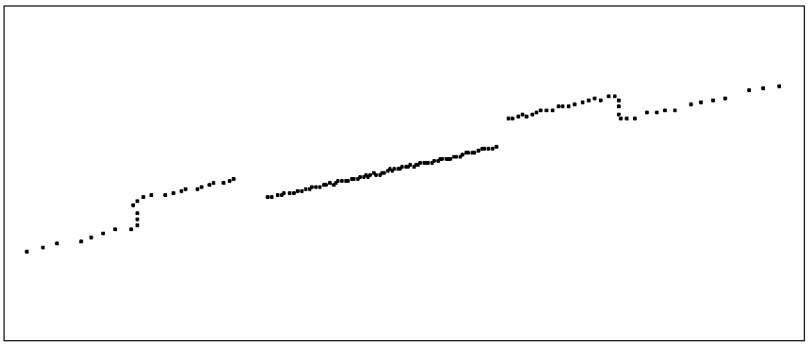


Figure 7. Raw Laser Points.

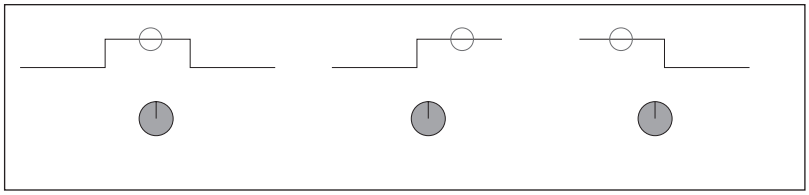


Figure 8. The Three Valid Elevator Patterns.

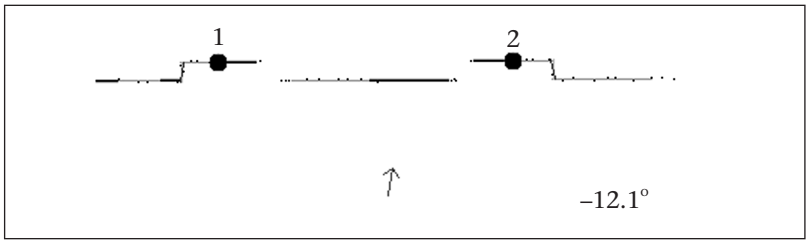


Figure 9. The System, Fully Settled, with Two Elevators Discovered.

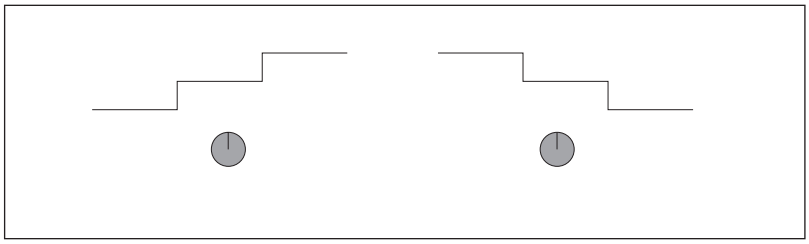


Figure 10. Two Invalid Elevator Patterns.

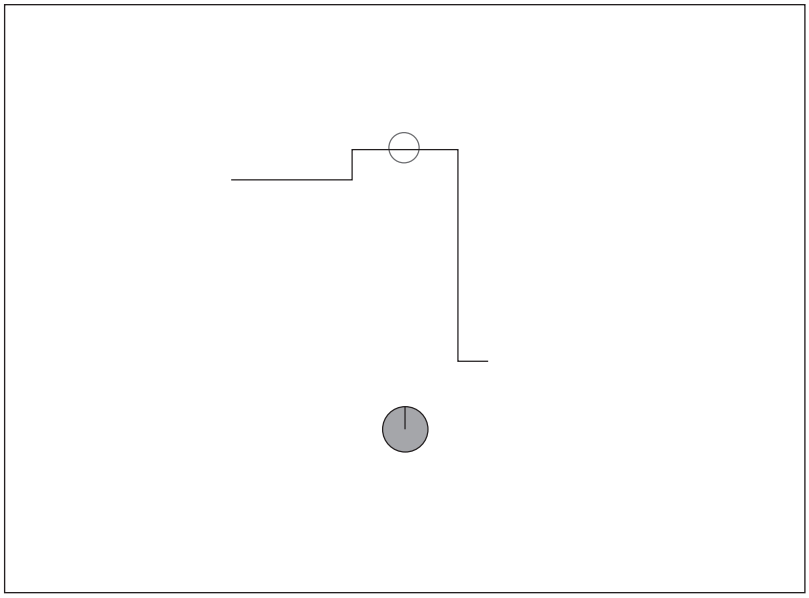


Figure 11. The Unusual Pattern at the Challenge.

sensor, based on an electronic altimeter, to determine which floor the robot is on.

Finding the Registration Booth

Once GRACE reached the registration area (see Getting to the Registration Area), the next task was to move up to the registration desk, which involved two related subtasks: (1) searching for and visually acquiring the sign indicating the registration desk and (2) servoing to the desk guided by a visual fix on the sign. The standard registration signs used at the Shaw Convention Centre, which were LCD displays, were too small and too dim to be seen by the robot's cameras. Therefore, we provided our own bright pink registration sign (figure 12).

The Swarthmore vision module (SVM) (Maxwell et. al. 2002) provided the vision software capabilities used for this task. SVM is a general-purpose vision scheduler that enables multiple vision operators to run simultaneously with differing priorities yet maintain a high frame rate. It also provides tightly integrated control over a pan-tilt-zoom camera, such as the Canon VC-C4 that was used on GRACE.

The SVM library includes a number of vision operators, one of which (the color blob detector based on histograms) was used to find the pink sign above the registration desk. In addition, each vision operator can function in as many as six different modes, including the PTZ_SET and LOOK_AT modes that were used with GRACE. The PTZ_SET mode allows software external to the SVM to set the position of the camera by designating pan, tilt, and zoom parameters. SVM does not independently move the camera in this mode. In the LOOK_AT mode, SVM is given the three-dimensional (3D) location of the camera and object to be tracked and sets the camera to point at the object. If the vision operator finds the object, SVM moves the camera to track it within a limited region around the designated location. The software for servoing GRACE to the registration desk, including the interface to both SVM and the lower-level locomotion software, was written using TDL.

Because of the configuration of the registration area at the Shaw Centre, GRACE was approximately 15 to 20 meters from the registration desk when it first reached a position to be able to see the registration sign. The first phase of the task, searching for and finding the sign, was complicated by the configuration of the registration area. Although the pink sign was 0.5 by 1.0 meters in size and designed to be relatively easy to find, at a distance of 15 to 20 meters, with the camera's zoom set to the widest angle (45-degree field of view), the sign was

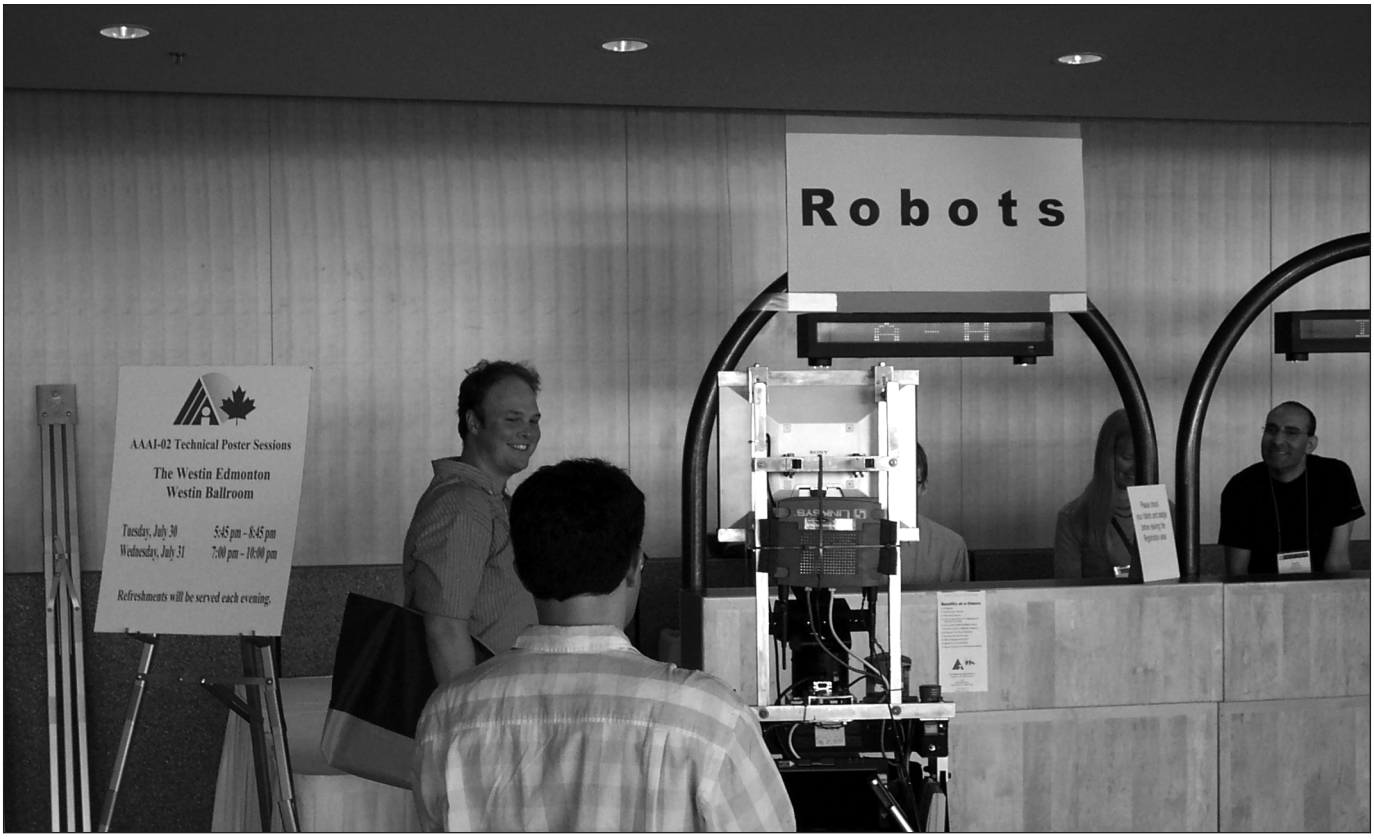


Figure 12. The Robot Registration Sign.

only a few pixels in size and nearly impossible for SVM's blob-detection operator to find. To achieve more robust sign detection, we increased the zoom (narrowing the field of view to 5 degrees), resulting in a very meticulous, but slow, search process. During this phase, SVM was used in PTZ_SET mode, giving full control of the camera to the TDL code. The shifting light levels in the registration area, because of time and weather changes, also caused some difficulties. Histograms for the pink sign trained at a certain time of day often failed several hours later. To ameliorate this problem, we trained the histograms immediately before the start of the challenge.

Once the registration sign was found, an approximate distance to the sign was calculated based on the blob elevation measure provided by SVM. This measurement, in turn, was used to calculate the 3D location of the sign in the robot's global coordinate frame. At this point, the robot oriented itself to the sign and began moving toward the registration desk. The blob-detection operator was now changed to LOOK_AT mode, providing robust tracking of the sign during movement. SVM provided updates on the position of the sign in the pan-tilt frame of the camera, which were then translated into global coordinates by the TDLcode, providing both sign and robot location updates to SVM as well as correcting the movement of the robot. The TDL code also adjusted the zoom used by SVM—as GRACE's distance to the sign decreased, the field of view of the camera was increased to maintain the entire sign within the image, thereby reducing the chance of losing the sign and producing more accurate estimates of its location. This part of the task was considered completed when GRACE reached a distance of two meters from the desk.

Standing in Line

Once GRACE was near the registration desk, it proceeded to register. First, however, it waited in line (if there was one) like any polite conference attendee. GRACE uses a combination of an understanding of personal space and range information to stand in line. GRACE uses the concept of personal space to understand when people are actually in line rather than when they are milling around nearby. People standing in line will typically ensure that they are close enough to the person in front of them to signify to others that they are in line yet maintain a minimum socially acceptable separation distance. GRACE also uses this information to

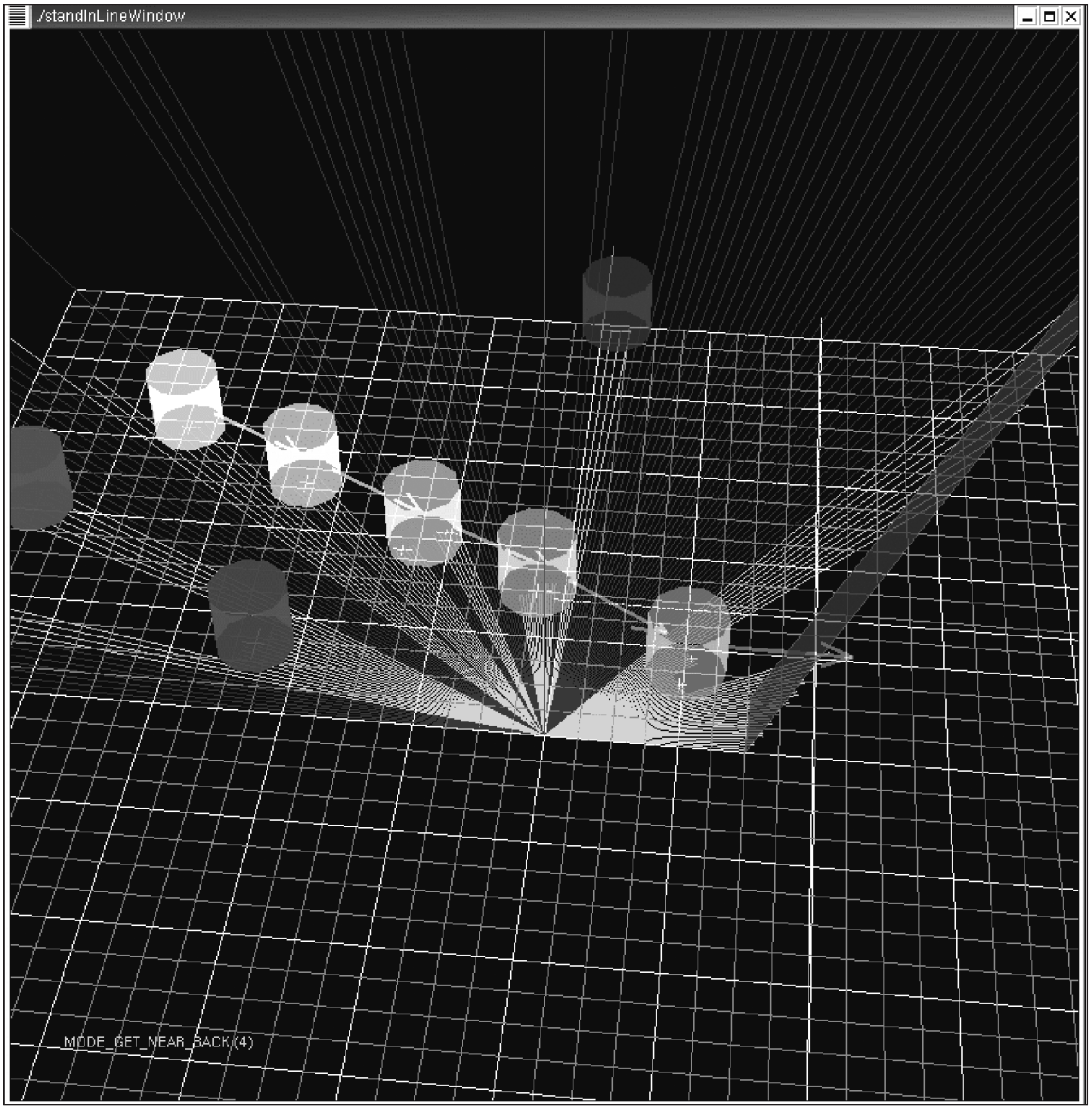


Figure 13. GRACE*'s Perception of People in Line.*

ensure that once in line, it does not make others feel uncomfortable by getting too close to them. The algorithm is based on earlier work using stereo vision for detecting lines (Nakauchi and Simmons 2002).

GRACE uses the SICK scanning laser range finder to identify people and walls. Before each movement, a laser scan is performed. Clusters in the range data are grouped into three categories: (1) those that might be people, (2) those that are likely walls, and (3) those that are other (figure 13). This classification is based on the shape of the cluster. To identify people, the algorithm looks for a small cluster of data points (with a spread of less than approximately 50 centimeters) or a pair of small clusters close together. This simple heuristic incorrectly classifies a variety of objects that are not people as people, but these "false positives" are generally irrelevant in the context of standing in line to register for a conference.

If a cluster is too big to be a person and the points in the cluster fall approximately along a line, the cluster is considered a wall. Occlusions in the range data (figure 13) are compensated for by comparing wall clusters to one another to determine if a single wall segment can explain them. If such is the case, then these clusters are combined to provide a better estimate of the orientation and location of the walls.

The stand-in-line algorithm assumes that GRACE starts near the registration desk and that the closest "wall" is the front of the desk. Once the closest wall has been found, GRACE rotates away from the desk and searches for the nearest person standing close to the registration desk. This person is considered to be the "head

of the line." Once the head of the line has been identified, the algorithm attempts to chain nearby people together using the notion of personal space. Those that are too far from the person in front of them, or those who are not approximately behind someone in line, are not considered to be in line. Once the line is found, GRACE moves toward the back of the line, intermittently checking for more people in line. Once at the back of the line, GRACE moves to a position behind the last person. At this point, GRACE only considers the person immediately in front of it, maintaining the personal space between the robot and this person. Once near the registration desk, GRACE maintains a standoff distance until the person in front leaves. When there are no more people in front of GRACE, it drives to a set distance from the registration desk and then begins to register.

Registering

The objectives for this subtask were to develop an interaction system that was robust enough so that a (relatively) untrained person could interact with it and to present an interface that was natural enough so that the registrar and observers could interact with GRACE at least somewhat as they would with a human. The specific task was for GRACE to obtain all the various registration paraphernalia (bag, badge, proceedings) as well as find out the location and time of its talk.

Figure 14 illustrates the data and control flow for a typical interaction cycle with the robot. A wireless microphone headset is used to acquire speech, which is then converted to text by VIAVOICE. VIAVOICE has the ability to read in a user-specified BNF (Backus-Naur form)-style grammar, which it then uses to assist in speech disambiguation. In fact, it will only generate utterances that are valid under the loaded grammar. Obviously, there is an inverse relationship between the size of the grammar and the recognition accuracy of VIAVOICE (when presented with valid utterances). We built our own grammar to cover all the potential utterances we could think of within the given scope. Because the breadth of interaction involved in performing the registration task is rather limited, we were able to achieve satisfactorily accurate recognition.

After recognizing an utterance, VIAVOICE transmits it across the network as a string. NRL developed a module, called UTT, which listens for transmissions from VIAVOICE and re-broadcasts them over IPC as "utterance" messages. UTT also has a text-based input mode, which is useful for debugging. The text strings are then parsed by the UTT2SIGNAL program. UTT2SIGNAL performs the same basic function as NAUTILUS but is significantly more simple and specialized. UTT2SIGNAL is based on a Bison parser that was hand generated from the VIAVOICE BNF grammar. It distills the utterances down to the primitives that we need to drive our interaction and transmits the appropriate signals to the "expression" process (see later discussion). In addition, UTT2SIGNAL is responsible for dispatching any raw information gleaned from the utterances to the appropriate process. For

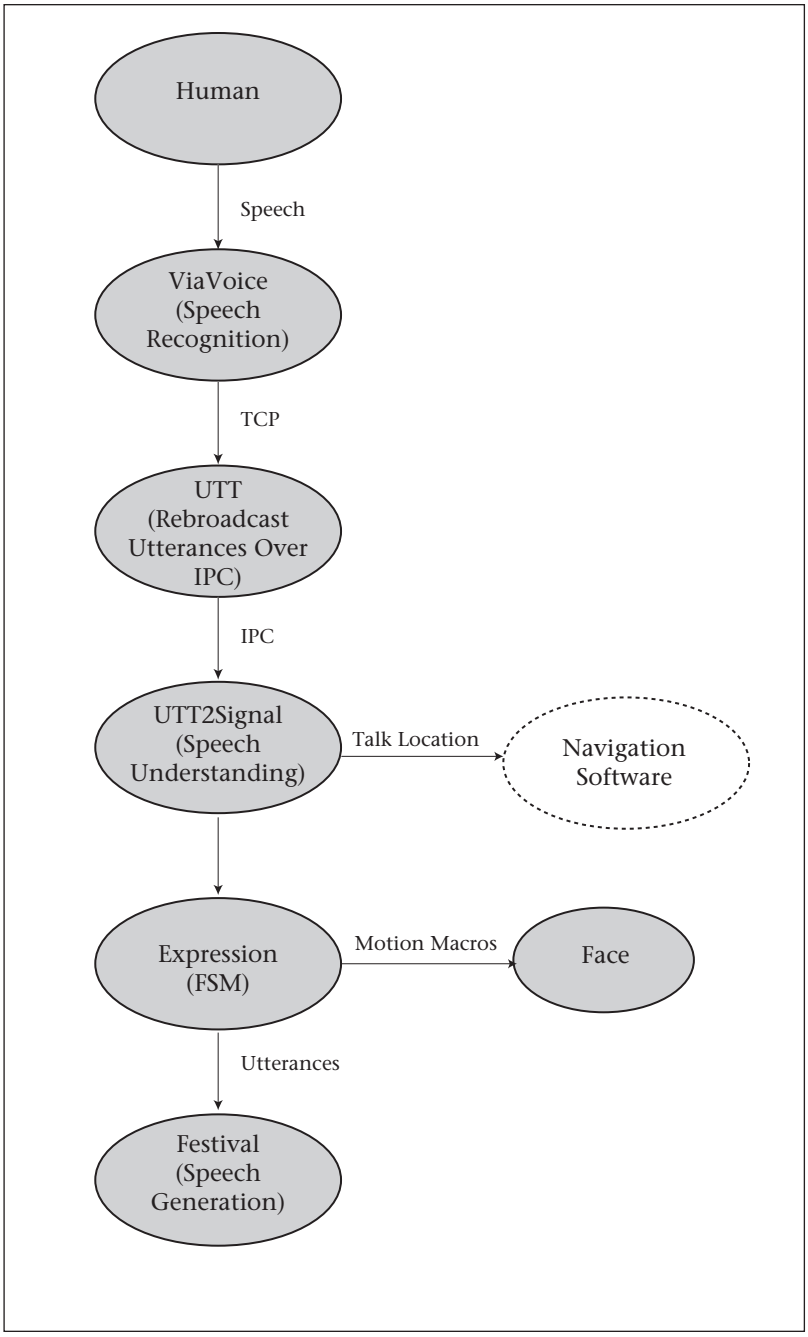


Figure 14. Information Flow for the Registration Desk Task.

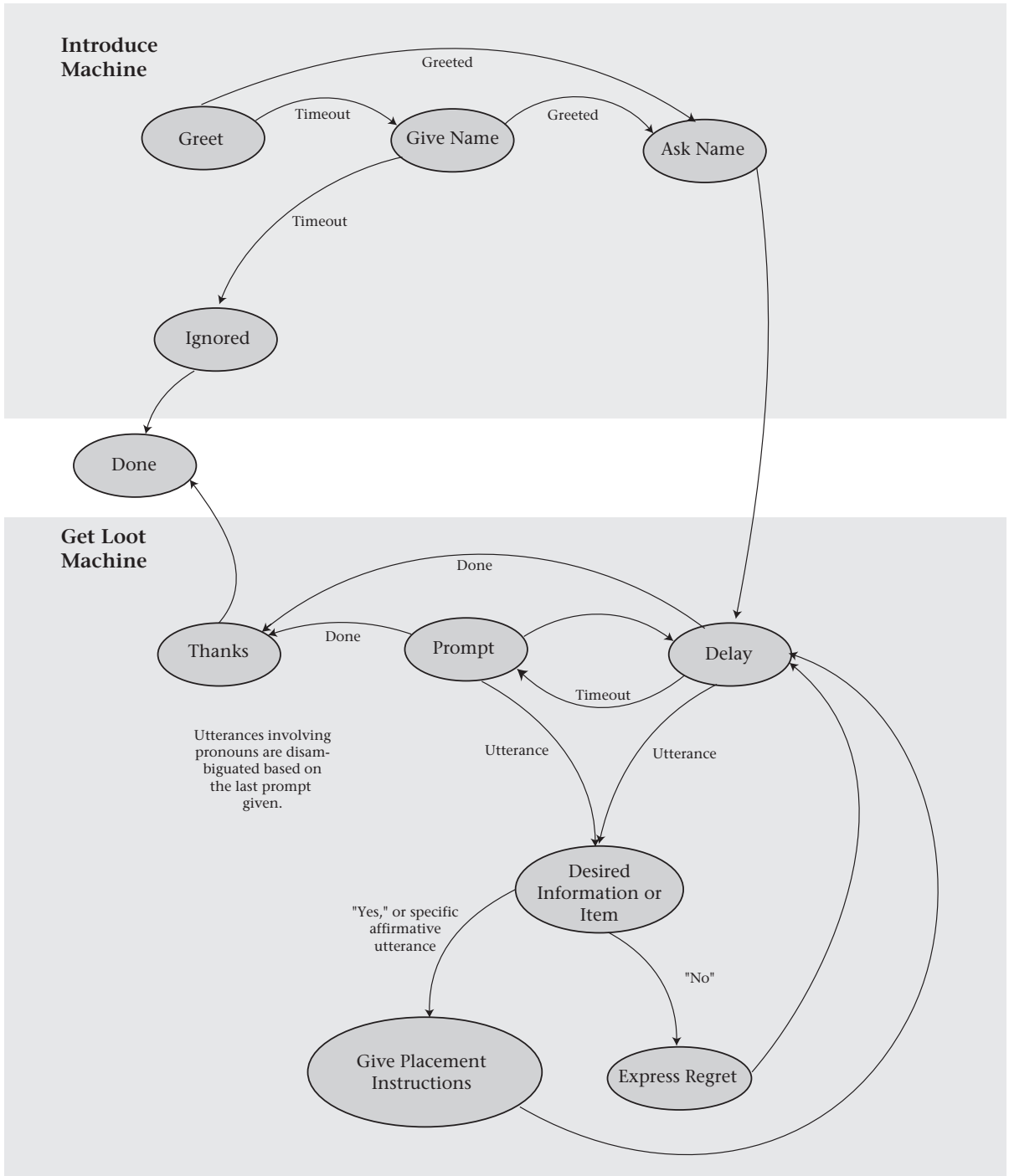


Figure 15. Simplified Finite-State Machine for the Registration Task.

example, if the registrar tells GRACE the location of its talk, UTT2SIGNAL informs the navigation software of this same location.

The "expression" process controls the computer-animated face and the FESTIVAL speech-generation software. Users write interaction scripts that include facial expressions, quoted text, pauses, conditional operators, choice operators, and most basic math and logic operations. The scripting language allows the definition of macros, which include basic face movements, utterances, and nonface primitives (such as pauses). Even more powerful is the ability to create and execute hierarchical finite-state machines (FSMs) (figure 15). The FSMs can execute actions when entering a state and can transition based on signals received from other processes (for example, UTT2SIGNAL, hence the name). Figure 16 shows a small sample of the script used for the registration task.

Because UTT2SIGNAL abstracts out the actual parsing, the FSM can concentrate on the content, which decreases its complexity. In addition, execution time scales well with the size and number of FSMs. In the future, this scalability will allow more complex interactions to be driven without unreasonable computational requirements.

GRACE's face (figure 17) is one of the most important aspects of its ability to interact with humans. It is used for both emotional expression and simple gestures because GRACE lacks any conventional manipulators. The face is based on an implementation of the simple face in Parke and Waters (1996). It incorporates a muscle-level model of face movement to allow semirealistic face motions. It accepts muscle and simple movement commands from expression; macros of these commands are built up within the "expression" process to allow easy access to complicated expressions or gestures.

Last, but not least, is GRACE's ability to generate speech. We use a version of FESTIVAL that was modified to enable it to generate phonemes for a given utterance, which are then processed to extract lip-synching information. FESTIVAL performed admirably, overall, with two notable exceptions: (1) it tends to speak in a monotone and (2) it cannot handle acronyms. Although it is possible to embed pitch changes in strings sent to FESTIVAL, this capability was too labor intensive to take advantage of this year and does not tend to produce convincing speech in any case. Likewise, it is possible to embed phonetic pronunciations to deal with utterances such as "AAAI."

There were a number of small, persistent problems with the interaction. First, VIAVOICE had trouble with short utterances, often misinterpreting them as numbers. Because an utterance of just numbers was parsed as a statement of the time of GRACE's talk, this approach could cause some confusion. However, GRACE was able to recover from such mistakes because of the structure of the driving FSM.

The other problem had to do with the disambiguation of pronouns and other generic statements. GRACE disambiguates such statements as "here you go," "no," or "you have it" based on the latest prompt that it gave (that is, what state of the FSM it is currently in). However, if GRACE prompted the registrar, and the registrar began to respond, but VIAVOICE did not complete recognizing the utterance until after GRACE had timed out and begun the next prompt, GRACE would believe that a nonspecific statement was about the new prompt, even if it had only said a syllable or two of it. This timing issue obviously caused some problems because the potential existed for its belief of the state of the world to get out of sync with reality, resulting in very unnatural interaction.

Navigating to the Talk

After registering, the challenge robots are allowed to use a map to navigate in the building. Ideally, the robots would actually read the map given to them. GRACE, however, used a map that it had built previously and had saved on disk. The map was used to help GRACE make its way from the registration desk to the talk venue. The map-based navigation task comprised three main technologies: (1) map building, (2) localization, and (3) navigation control.

The evening prior to the challenge event, GRACE was driven around the convention center. During this time, time-stamped odometry and laser range data were recorded to file. These data were then used to build a map through a process called *scan matching* (Lu and Milios 1997). The implementation of our scan-matching algorithm was adapted from a software package provided by Dirk Hähnel of the University of Freiburg (Hähnel et. al. 2002). Generating a map from laser and odometry data is largely an automated process, although our implementation also allows the user to correct misalignments after the scan-matching process. The output of the map-building process is an occupancy grid map, shown in figure 18. This map is 89.4 by 10.8 meters, with a resolution of 10 centimeters for each grid cell. The black pixels represent regions of space with a high probability of occupancy, such as walls and chairs. Similarly, the white areas are regions of space with a low probability of occu-

```
# Example of expression definition
#   Expression definitions are of the form
# DEFINE expressionName
# { say("<utterance>")
#   [one or more expression macros]
#   [lip synching macros]
# }
#
# For example:

DEFINE badgeYesPrompt
  { say("May I have my badge please?")
    [dhappy2]
    [pause(0.129) mm me mi ma mm mi ma msh mp me pause(0.079) msh mn]
}

# Example of DFA / FSM

# Inclusion of other FSM and expression definition files is
#   allowed for maximum flexibility
include "register.fsm"
include "mutter.pho.expr"

# Define the initial and final states of a FSM
BEHAVIOR-MACHINE MutterMachine
  initial MM_Enter
  final MM_Final

BEHAVIOR MM_Enter
  # Transition immediately if either of these signals is received,
  #  even interrupting speech in progress
  transition interrupted "speech:reset" MM_Final
  transition interrupted "control:stopMutter" MM_Final
  perform
   [# Serialize everything in []'s
    # First, choose something to say
     CHOOSE(
        mutter1,
       mutter2,
       mutter3),
     pause(2),
     removeTextBubble,
     slowNormal,
     smiley,
     # Then, choose how long to wait
     CHOOSE(
        pause(5),
        pause(15),
        pause(30))
   ]
   # Finally, do this all over again
   # This transition fires only when the preceding perform clause
   #  has completed
   transition MM_Enter

# There are no transitions out of this node, thus signaling the
#  termination of the FSM
BEHAVIOR MM_Final
  perform slowNormal
```

Figure 16. Sample Expressions and Finite-State Machines for the Registration Desk Task.

Figure 17. GRACE*'s* FACE*.*

pancy. Not shown in this image are regions of space where no data could be collected (that is, behind walls).

GRACE uses a probabilistic approach to localization called Markov localization. The localizer estimates a probability distribution over all possible positions and orientations of the robot in the map given the laser readings and odometry measurements observed by the robot. This probability distribution is approximated using a particle filter (Thrun et. al. 2000). GRACE is initialized with an approximate starting position, and the distribution of particles evolves to reflect the certainty of the localizer's position estimate.

As GRACE moves, the probability distribution is updated according to

$$p(s_i) = \eta \cdot p(o_i|s_i) \int p(s_i|s_{i-1}, a_{i-1}) p(s_{i-1}) ds_{i-1}$$

where s_i is the pose at time i, a_{i-1} the last action, and o_i the last observation.

Navigation was performed using a two-level system. The low-level system uses the lane-curvature method (Ko and Simmons 1998) to convert commands in the form of directional headings to motor velocity commands. The high-level planner consists of an implementation of a Markov decision process planner (Burgard et. al. 1998; Konolige 2000). The planner operates by assigning a positive reward to the goal location and a negative reward to poses close to obstacles. The planner uses value iteration to assign a value to each cell; this value

Figure 18. Map Built by GRACE *of the Shaw Convention Centre.*

corresponds to the future expected reward of each cell, as in the following equation:

$$V(s_i) = \max_a \left(R(s_i) + \gamma \sum_{j=1}^{|S|} V(s_j) \sum_{k=1}^{|A|} p(s_j \mid \pi(a_k \mid s_i), s_i) \right)$$

where $R(s_i)$ is the immediate reward of robot pose s_i, and $V(s_i)$ is the expected reward to be maximized. The planner extracts the maximum-likelihood path by choosing from the start state (the current pose of the robot as given by the localizer) successive states that maximize the expected reward. The directional command passed to the low-level controller is just the direction of the neighboring state with the highest-expected reward.

During execution of the planned path, the planner also integrates sensor information, based on the current pose estimate from the localizer, to make changes to the map. Thus, the planner is allowed to compensate for small errors in localization and changes to the environment that could invalidate certain paths.

Giving the Talk

Once GRACE navigated to the lecture area (in the Exhibition Hall), it gave a talk about the technologies that it comprises (figure 19). GRACE's talk-giving system is an attempt to scale behavior-based architectures directly to higher-level cognitive tasks. The talk giver combines a set of behavior-based sensory-motor systems with a marker-passing semantic network, a simple parser, and an inference network to form an integrated system that can both perform tasks and answer questions about its own ability to perform these tasks. It interfaces with the computer-animated face and FESTIVAL speech-generation systems to do the actual presentation.

The talk system is structured as a parallel network of logic gates and finite-state machines.

Figure 19. GRACE *Gives a Talk.*

Inference rules in the system are compiled into a feed-forward logic network, thus giving it circuit semantics: The input of the network monitor the truth values of premises as generated by the sensory systems, and the output of the network track the truth values of conclusions in real time as the premises change. In effect, the entire rule base is rerun from scratch to deductive closure at sensory frame rates. Although this approach sounds inefficient, the rule engine can run a base of 1000 Horn rules with 10 conjuncts each, updating at 100 Hertz (100 complete reevaluations of the knowledge base a second), using less than 1 percent of the central processing unit. Using a generalization of deictic representation called ROLE PASSING (Horswill 1998), the network is able to implement a limited form of quantified inference—a problem for previous behavior-based systems. Rules can be quantified over the set of objects in short-term memory, provided they are restricted to unary predicates (predicates of one argument).

The talk-giving system implements *reflective knowledge*—knowledge of its own structure and capabilities—through two mechanisms: (1) a marker-passing semantic network provides a simple mechanism for long-term declarative memory and (2) role passing allows variables within inference rules to be bound to behaviors and signals within the system. The first mechanism allows the system to answer questions about its own capabilities, and the second mechanism allows it to answer questions about its current state and control processes.

The talk-giving system can follow simple textual instructions. When a human issues a command such as "drive until the turn," its simple parser, which is formed as a cascade of finite-state machines, examines each individual word, binding the appropriate words to the appropriate roles. In this case, the parser binds the drive behavior to the role activity and the turn? sensory signal to the role destination. When it detects a stop (for example, a pause), it triggers the handle-imperative behavior, which implements the rules:

If the signal bound to destination is false, activate the behavior bound to activity.

If destination is bound to a sensory signal and that signal is true, deactivate activity and the system.

If activity deactivates itself, also deactivate the handle-imperative behavior.

Because this behavior is parameterized by other behaviors, we call it a higher-order behavior in analogy to the higher-order procedures of functional programming languages. Other examples are the *explain behavior,* which walks a subtree of the semantic network to produce a natural language explanation of the behavior, and the *demo behavior,* which both explains and runs the behavior. Role passing and higher-order behaviors are easily implemented using parallel networks of gates and FSMs, making them a natural choice for the kind of distributed, parallel processing environments often found on mobile robots. They are implemented in GRL, a functional programming language for behavior-based systems that provides many of the amenities of Lisp and statically compiles programs to a network of parallel FSMs.

To give a talk, GRACE uses the Linksys wireless connection to a laptop to open a POWERPOINT presentation, reads the text of each bullet point, and uses keyword matching to find an appropriate node in its semantic network. It uses a novel distributed representation of a discourse stack to resolve ambiguities, using only parallel marker-passing operations. Having determined the node to which the bullet point refers, GRACE uses spreading activation to mark the subtree rooted at the selected node as being relevant. It then discusses the topic by continually selecting and explaining the highest priority relevant, unexplained node. Priorities are computed offline using a topological sort so that if topic *A* is required to understand topic *B*, *A* will always have higher priority.

Continually reselecting the highest-priority, relevant, unexplained node using circuit semantics gives GRACE the ability to adjust its pattern in response to unexpected contingencies. Although the current system doesn't make much use of this capability, we intend to make extensive use of it in next year's system. If, for example, GRACE had to maneuver its way around a bystander in the process of demonstrating its navigation system, it might insert a digression about social interaction and the need to say "excuse me." When later in the talk, it came to the section on social interaction, it would realize it had already discussed the topic and simply make reference to its earlier discussion. It also allows the robot to cleanly respond to, and return from, interruptions without replanning. However, such topic shifts require the generation of transition cues such as "but first …" or "getting back to…." The talk code detects these abrupt topic shifts by tracking the current semantic net node, its parent node, and the previous node and parent. By comparing these nodes, the system can determine whether it has moved locally up, down, or laterally in the hierarchy or whether it has made a nonlocal jump to an unrelated node. It then generates the appropriate transition phrase.

The talk giver is far from fluent. It is not intended to demonstrate that behavior-based

systems should be the implementation technique of choice for natural language generation. Instead, it shows that parallel, finite-state networks are much more powerful than previously believed. Moreover, by implementing as much of a robot's control program as possible with these techniques, we get efficiency, easy parallelization, and flawless synchronization of the knowledge base with the environment.

Discussion and Summary

On Wednesday, 31 July, GRACE attempted the AAAI robot challenge in front of hundreds of interested onlookers and the media. GRACE successfully completed each of the subtasks described earlier with a minimal amount of extraneous human intervention. GRACE took about 60 minutes to travel from the entrance of the Shaw Convention Centre, down the elevator, to the registration desk, and then to the lecture area in the Exhibition Hall. This performance compares to about 20 minutes taken by the other entry that attempted the complete challenge—the *CoWorker* built by iRobot—but this robot was remotely teleoperated by a person in the convention center.

Although each of the subtasks was successful, and GRACE successfully completed an end-to-end run, each subtask also demonstrated the need for improvement. Probably the most critical problem was based on our use of VIAVOICE for speech recognition. VIAVOICE has trouble with background noise and stress in the speaker's voice. Although we have found that someone who has worked regularly with VIAVOICE can achieve high recognition rates using our large vocabulary and grammar, new users in stressful situations can have greatly reduced recognition rates. A Ph.D. student working at NRL for the summer to work on the GRACE project did the interaction during the challenge. With each misunderstood utterance, the level of stress in the student increased (particular with the very large crowd of onlookers and press), resulting in yet lower recognition rates. To try and remedy this, we are in the process of evaluating Sphinx for speech recognition.[9] More robust speech recognition might also enable us to move to an on-robot microphone system, which would eliminate the need for the speaker to don a wearable microphone and would also enhance GRACE's appearance as an independent entity and enable random interaction.

Although the human-robot interaction (aside from the speech recognition) worked relatively well, there were areas for improvement. For example, gesture recognition, which works on the NRL robots, was not successfully integrated in time for GRACE. As a fallback position, a PDA device was programmed to allow the human to "point" a direction on its screen. However, this interface failed to start properly at the beginning of GRACE's run. Without the ability for the human to give gestures, the resulting interaction was closer to "verbal teleoperation." We expect to have full gesture capability in 2003.

In addition, NRL has developed an ability to talk about semantic entities in the environment (for example, "turn left down the next corridor"), but the ability to recognize these features is not yet integrated into GRACE. These capabilities would make interaction much more natural.

For the elevator-riding task, the robot needed to have a person hold the elevator doors open to give it time to enter and exit before the doors closed because, in part, the robot did not recognize changes to the environment fast enough. Also, the robot did not have any way of determining which floor it was on (we are working on this by developing an electronic altimeter—see Riding the Elevator).

Visual servoing to the registration desk suffered from several problems. First, as described in Finding the Registration Booth, changes in lighting could cause the recognition algorithm to fail; so, the system had to be retrained on a periodic basis. Second, when the robot was far away, the sign appeared too small to readily be identified, but zooming in gave a very small field of view, which slowed the search for the sign considerably. To deal with this problem, we are considering a multiscale approach, where the robot first does a coarse scan at a wide field of view and then checks possible sign locations more thoroughly by zooming in. Finally, if the robot moved quickly, the tracker often lost sight of the sign, which can also probably be addressed by adjusting the zoom.

During testing, the standing-in-line code was very reliable. During the challenge itself, the robot barged into line, nearly hitting one of the judges. The cause was traced to a bug in the software that determined the robot's trajectory to the end of the line. The software worked in many tests, but later it was determined that it only worked for lines of one or two people (the maximum we had tested on), but at the challenge there were five people in line. Needless to say, that bug has since been fixed. The task of registering demonstrated problems with VIAVOICE, as described earlier. Also, that task used a different grammar from the "getting to the registration area" task. During the challenge, we forgot to load the correct grammar, which

meant that the robot had very little chance of interacting correctly. Fortunately, the incorrect grammar was noticed, and corrected, part way through the task.

The navigation part of the task suffered a bit from getting lost. The causes were twofold: (1) the environment had changed significantly from when the map was built the night before (extra tables were set up for food) and (2) there were hundreds of people around the robot, making it hard for the sensors to see the walls and other static structures that had been mapped. Unfortunately, some human intervention was needed to relocalize the robot. We need to look much more carefully at how to do map-based navigation in environments that are very different from when the map was first made. Finally, the talk-giving task worked flawlessly. For next time, however, we plan to have the robot demonstrate various aspects of itself, which is currently supported in the talk-giving software, but there was not enough time to develop the demonstrations themselves and integrate them into the talk.

Our plans for the 2003 challenge are threefold. First, we will work to make the current capabilities much more robust. Second, we will integrate the capabilities more tightly. In particular, we will have the robot itself determine when to transition between subtasks. Third, we will add new capabilities. We intend to have vision-based and stereo-based people detection and tracking, people following, gesture recognition, name-tag reading, and face recognition. We plan to incorporate capabilities for the robot to "schmooze" with other participants and answer its own questions after the talk. We would like to have the robot perform its own crowd control. In the hardware domain, it would be desirable to add more physical flexibility to GRACE's face, such as putting the screen on a pan-tilt unit. Finally, there is the possibility of bringing another robot and have a team try to attend the conference. Imagine two attendees who arrive together, looking together for the registration area, or one that arrives earlier and meets the second near the entrance and tells it what it has learned about the location of the registration desk. We are also considering having one robot handle crowd control for the second robot!

Our biggest lesson learned was the amount of work required to achieve interaction among the different modules. The bulk of our work this first year was in getting the interfaces defined and working. However, we believe that we have to go much further next year. In particular, we had several failures occur because we had to manually start software when moving from one part of the challenge to the next. We will be developing a program that will automatically start each module, which should result in fewer human errors.

Although we did accomplish much this year, we are looking forward to adding a significant amount to GRACE for next year. In addition to the automatic starting of processes, we expect to have tighter interaction among components, different and more robust human interactions, gesture recognition, better recognition of humans using multiple techniques, and possibly even the ability for the robot to demonstrate itself during its talk and answer simple questions.

Notes

1. www.cs.cmu.edu/~IPC.
2. www.ibm.com/software/speech.
3. www.cstr.ed.ac.uk/projects/festival.
4. www.cs.cmu.edu/~IPC.
5. www.ibm.com/software/speech.
6. www.cstr.ed.ac.uk/projects/festival.
7. www.cs.cmu.edu/~TDL.
8. www.cs.utexas.edu/users/kuipers/AAAI-robot-challenge.html.
9. www.speech.cs.cmu.edu/sphinx.
10. www.cs.cmu.edu/~TDL.

References

Burgard, W.; Cremers, A. B.; Fox, D.; Hähnel, D.; Lakemeyer, G.; Schulz, D.; Steiner, W.; and Thrun, S. 1998. The Interactive Museum Tour-Guide Robot. In Proceedings of the Fifteenth National Conference on Artificial Intelligence, 11–18. Menlo Park, Calif.: American Association for Artificial Intelligence.

Hähnel, D.; Schulz, D.; and Burgard, W. 2002. Map Building with Mobile Robots in Populated Environments. Paper presented at the Conference on Intelligent Robotics and Systems, 30 September–4 October, Lausanne, Switzerland.

Horswill, I. 1998. Grounding Mundane Inference in Perception. *Autonomous Robots* 5(1): 63–77.

Ko, N. Y., and Simmons, R. 1998. The Lane-Curvature Method for Local Obstacle Avoidance. Paper presented at the Conference on Intelligent Robotics and Systems, 12–18 October, Vancouver, Canada.

Konolige, K. 2000. A Gradient Method for Real-Time Robot Control. Paper presented at the Conference on Intelligent Robotic Systems, 30 October–5 November, Takamatsu, Japan.

Lu, F., and Milio, E. 1997. Globally Consistent Range Scan Alignment for Environment Mapping. *Autonomous Robots* 4(4): 333–349.

Maxwell, B. A.; Fairfield, N.; Johnson, N.; Malla, P.; Dickson, P.; Kim, S.; Wojtkowski, S.; and Stepleton, T. 2002. A Real-Time Vision Module for Interactive Perceptual Agents. *Machine Vision and Applications* 14(1): 72–82.

Nakauchi, Y., and Simmons, R. 2002. A Social Robot That Stands in Line. *Autonomous Robots* 12(3): 313–324.

Parke, F., and Waters, K. 1996. *Computer Facial Animation.* London: A. K. Peters.

Perzanowski, D.; Schultz, A. C.; and Adams, W. 1998. Integrating Natural Language and Gesture in a Robotics Domain. In Proceedings of the International Symposium on Intelligent Control, 247–252. Washington, D.C.: IEEE Computer Society.

Perzanowski, D.; Schultz, A. C.; Adams, W.; Marsh, E.; and Bugajska, M. 2001. Building a Multimodal Human-Robot Interface. In IEEE Intelligent Systems, 16–21. Washington, D.C.: IEEE Computer Society.

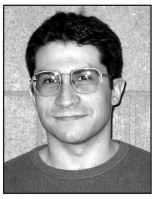
Perzanowski, D.; Schultz, A. C.; Adams, W.; Skubic, W.; Abramson, M.; Bugajska, M.; Marsh, E.; Trafton J. G.; and Brock, D. 2002. Communicating with Teams of Cooperative Robots. In *Multi-Robot Systems: From Swarms to Intelligent Automata,* eds. A. C. Schultz and L. E. Parker, 185–193. Dordrecht, The Netherlands: Kluwer.

Simmons, R., and Apfelbaum, D. 1998. A Task Description Language for Robot Control. Paper presented at the Conference on Intelligent Robotics and Systems, 12–16 October, Vancouver, Canada.

Thrun, S.; Fox, D.; Burgard, W.; and Dellaert, F. 2001. Robust Monte Carlo Localization for Mobile Robots. *Artificial Intelligence* 128(1–2): 99–141.

Wauchope, K. 1994. EUCALYPTUS: Integrating Natural Language Input with a Graphical User Interface. Technical Report NRL/FR/5510-94-9711, Naval Research Laboratory, Washington, D.C.

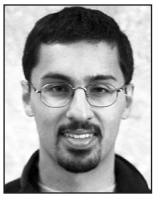
Reid Simmons is a principal research scientist in robotics and computer science at Carnegie Mellon University. His work focuses on developing reliable, autonomous robots, which includes research in robot architectures; probabilistic planning; and reasoning, formal verification, robot navigation, multirobot coordination, and human-robot social interaction. Simmons has been involved with the AAAI Robot Competition and Challenge since its inception. His e-mail address is reids@cs.cmu.edu.



Dani Goldberg received his Ph.D. and M.S. from the University of Southern California in 2001 and 1999 and his B.A. from Brandeis University in 1996. He is currently a postdoctoral fellow at the Robotics Institute, Carnegie Mellon University, where his work involves multirobot coordination and control. His e-mail address is danig@cs. cmu.edu.

Adam Goode is a research programmer in the ACT-R Research Group and a Masters' student in the Human-Computer Interaction Institute at Carnegie Mellon University. He earned his bachelor's degree in computer science and psychology in the Minds and Machines Program at Rensselaer Polytechnic Institute. His e-mail address is agoode@andrew.cmu.edu.

Michael Montemerlo is a graduate student in the robotics Ph.D. program at Carnegie Mellon University in Pittsburgh. His research interests include simultaneous localization and mapping (SLAM) and mobile robot navigation. He is a coauthor of the CARMEN mobile robot control software suite. His e-mail address is mmde@cs. cmu.edu.



Nicholas Roy is a graduate student in the robotics Ph.D. program at Carnegie Mellon University in Pittsburgh. His research interests include probabilistic planning models and machine learning. He is a coauthor of the CARMEN mobile robot control software suite. His e-mail address is nickr@ri.cmu.edu.

Brennan Sellner is a Ph.D. student at the Carnegie Mellon Robotics Institute, Carnegie Mellon University, in Pittsburgh. His primary research is currently cooperative manipulation with heterogeneous manipulators.

Chris Urmson is a Ph.D. student at Carnegie Mellon University in Pittsburgh, Pennsylvania. His research focuses on the development of real-time kino-dynamic planning techniques for mobile robots. His e-mail address is curmson@ri.cmu .edu.

Bruce Maxwell is an assistant professor of engineering at Swarthmore College. He received a B.A. in political science and a B.S. in engineering with a concentration in computer science from Swarthmore College, an M.Phil. from Cambridge University, and a Ph.D. in robotics from the Robotics Institute at Carnegie Mellon University. His primary research interests are computer vision, robotics, computer graphics, data mining, and visualization.

Alan Schultz is the head of the Intelligent Systems Section at the Navy Center for Applied Research in Artificial Intelligence at the Naval Research Laboratory in Washington, D.C. His work focuses on the area of autonomous systems, including human-robot interaction, multimodal interfaces, and learning and adapting in autonomous systems. Schultz has been active in robotics competitions, including the American Association for Artificial Intelligence conference and Botball and is a KISS Institute for Practical Robotics fellow. His e-mail address is schultz@aic.nrl.navy.mil.

Myriam Abramson received her B.Sc. and M.Sc. in computer science at George Mason University in

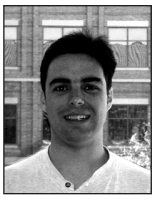
1984 and 1989, respectively, and is completing her Ph.D. in 2003 in the area of reinforcement learning. She has extensive experience in diverse areas of AI and machine learning. She has been a AAAI member since 1989. Her e-mail address is abramson@ccs.nrl. navy.mil.

William Adams received his B.S. in computer engineering from the Virginia Institute of Technology in 1991 and his M.S. in electrical and computer engineering from Carnegie Mellon University (CMU) in 1994. He is currently employed at the Naval Research Laboratory in Washington, D.C., working in many areas of mobile robotics, including computer vision; mapping; localization; planning; navigation; and multimodal robot interaction combining speech, gestures, and personal digital assistants. His e-mail address is adams@aic.nrl.navy.mil.

Amin Atrash received his B.S. in computer science in 1999 and is currently a graduate student at the Georgia Institute of Technology. He interned at the Naval Research Laboratory in the summers of 2001 and 2002. His research interests include robotics, machine learning, and planning.

Magda Bugajska is a computer scientist at the Navy Center for Applied Research in Artificial Intelligence. She received a B.S. in mathematics and computer science with a minor in AI and robotics from the Colorado School of Mines and is currently enrolled in an M.S. program in computer science at George Mason University. Her research interests include evolutionary computation, cognitive modeling, and robotics. Her e-mail address is magda@aic.nrl.navy.mil.

Michael Coblenz is a sophomore in the School of Computer Science at Carnegie Mellon University. He worked on the navigation from the door to the registration desk task in the challenge as a summer intern at the Naval Research Laboratory. His e-mail address is mcoblenz@andrew.cmu.edu.



Matt MacMahon is a Ph.D. student studying AI at the University of Texas at Austin. He has worked at Stanford University, the Austrian Research Institute for AI, NASA Johnson Space Center, and the Naval Research Laboratory. His research interests focus on the cognitive processes of spatial reasoning and natural language understanding in people and robots.

Dennis Perzanowski is a computational research linguist in the Intelligent Multimodal Multimedia Group at the Navy Center for Applied Research in Artificial Intelligence at the Naval Research Laboratory in Washington, D.C. His technical interests are in human-robot interfaces, speech and natural language understanding, and language acquisition. He received his M.A. and Ph.D. in linguistics from New York University. He is a member of the American Association for Artificial Intelligence and the Association for Computational Linguistics. His e-mail address is dennisp @aic.nrl.navy.mil.

Ian Horswill is an associate professor of computer science at Northwestern University. His research focuses on integrating high-level reasoning systems with low-level sensory-motor systems on autonomous robots. He received his B.Sc. from the University of Minnesota in 1986 and his Ph.D. in computer science from the Massachusetts Institute of Technology in 1993. His e-mail address is ian@northwestern.edu.

Robert Zubek is a Ph.D. candidate at Northwestern University, where he also received his previous degrees. Zubek's research interests include robotics and AI in computer entertainment. His e-mail address is rob@cs.northwestern.edu.

David Kortenkamp, Tod Milam, and Bryn Wolfe support NASA Johnson Space Center's (JSC) Automation, Robotics, and Simulation Division. Kortenkamp and Wolfe work for Metrica Inc., and Milam works for S&K Technologies. At JSC, the three of them are involved in a variety of robotics and AI projects in support of human space flight. Kortenkamp received his Ph.D. in computer science and engineering from the University of Michigan and his B.S. in computer science from the University of Minnesota. Wolfe received his B.S. in computer engineering from the University of Arizona and an M.S. in computer engineering from the University of Houston at Clear Lake. Milam has a B.S. in computer science from Drake University. The e-mail address for Bryn Wolfe is b.wolfe@ieee.org.