From: AAAI Technical Report WS-02-18. Compilation copyright © 2002, AAAI (www.aaai.org). All rights reserved.

GRACE:
An Autonomous Robot for the AAAI Robot Challenge

Reid Simmons, Dani Goldberg, Adam Goode, Michael Montemerlo,
Nicholas Roy, Brennan Sellner, Chris Urmson
Carnegie Mellon University

Alan Schultz, Myriam Abramson, William Adams, Amin Atrash,
Magda Bugajska, Michael Coblenz, Matt MacMahon, Dennis Perzanowski
Naval Research Laboratory

Ian Horswill, Robert Zubek
Northwestern University

David Kortenkamp, Bryn Wolfe, Tod Milam
Metrica, Inc.

Bruce Maxwell
Swarthmore College

1. Introduction

The AAAI Robot Challenge was established four years ago as a "grand challenge" for mobile robots. The main objectives of the Challenge are to (a) provide a task that will demonstrate a high level of intelligence and autonomy for robots acting in a natural, peopled, dynamic environment, (b) stimulate state-of-the-art robotics research to address this task, and (c) 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.

The Challenge task is to find the registration booth and register at the National Conference on Artificial Intelligence, interact with other attendees, and give a technical talk on itself in an assigned room, and at an assigned time. 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, that means that 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 in order to get current state-of-the-art robots to achieve the task.

There are a number of important technologies that are needed meet the Challenge. These 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, natural language generation, speech understanding, knowledge representation, and social interaction with people. While researchers have worked on all of these areas, to a greater or lesser extent, they all need further work to be robust in the environment that the Challenge specifies. In addition, the technologies have not been fully integrated with one another.

In August 2001, several of the authors agreed to join efforts to attempt the Challenge in its entirety. We had all been working on technologies related to the Challenge, and felt that by pooling our efforts we could do fairly 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 very difficult. Our first year goal, therefore, was to create an architecture and infrastructure that would enable us 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 multi-year effort, and that in subsequent years we would build on this year's robot system.

In email 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 that 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 Section 2) and common integration architecture (see Section 3). In the end, we achieved somewhere between option 2 and 3, with the robot successfully performing most of the major subtasks with little human interaction (see Section 4). In July 2002, we traveled to the National Conference on Artificial Intelligence at the Shaw Convention Centre in Edmonton, Alberta to take part in the Challenge.

2. Robot Hardware

1. The Robot GRACE

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 (see Figure 1). The sensors that come standard with the B21 include touch, infrared, and sonar sensors. Near the base is a SICK scanning laser range finder that provides a 180-degree field of view.

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 synthesizer, and understand responses using a wireless microphone headset (a Shure TC Computer Wireless transmitter/receiver pair) and commercial speech recognition software (IBM's ViaVoice).

GRACE runs all software on board. Two 500 MHz 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 for giving the talk).

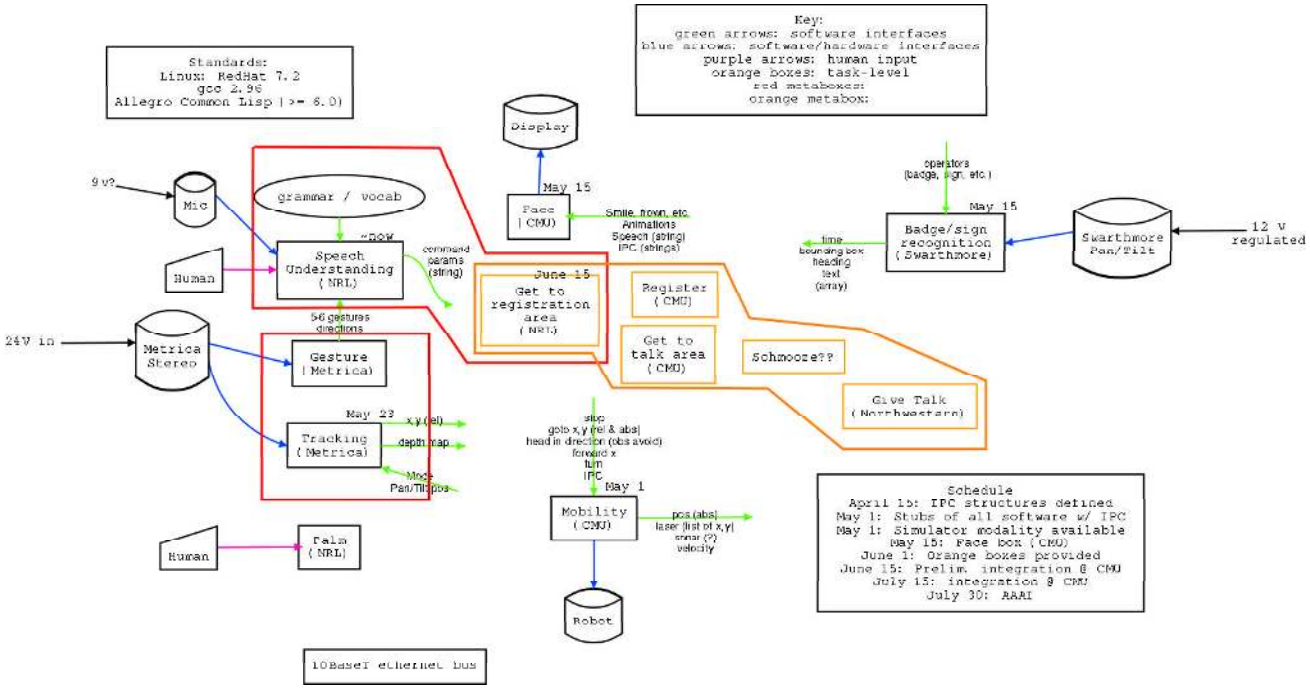
3. Software Architecture

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. Early on, we decided to integrate everything onto a common hardware platform (see Section 2), 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 platform, making subsequent development easier. In particular, Carnegie Mellon provided interfaces to the robot base (motion and localization), speech generation, and computer-animated face, the Naval Research Laboratory provided speech recognition and speech understanding interfaces, Swarthmore 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, Carnegie Mellon provided a simple graphical 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. While 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, the Naval Research Laboratory was responsible for the part of the Challenge from when the robot entered the conference center until it was near the registration booth, Carnegie Mellon was responsible for elevator riding, getting to the registration booth (using Swarthmore's vision system), registering for the conference, and navigating to the lecture area, and Northwestern was responsible for having GRACE give its talk. Figure 2 presents a high-level view of the software architecture and development responsibilities. Section 4 presents details of the task-level software.

To facilitate distributed development, and to simplify testing and debugging, the GRACE system was designed as a set of independent programs that communicated via message passing. The IPC package (www.cs.cmu.edu/~IPC) was chosen for (nearly all) communications, because of its expressiveness, ease of use, and familiarity by some of the teams (both Carnegie Mellon and Metrica have used IPC in the past). 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 (www.ibm.com/software/speech), the use of Allegro Common Lisp for the Nautilus speech recognition system, the use of Swig and Python for the elevator riding code. In addition, OpenGL, Perl and Festival (www.cstr.ed.ac.uk/projects/festival) were used for the computer-animated face and speech generation.

Finally, the computer-animated face and several 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



2. GRACE Software Architecture Diagram

monitoring, and exception handling (see www.cs.cmu.edu/~TDL and [Simmons & Apfelbaum, 1998]). A compiler translates TDL code into pure C++ code that includes calls to a domain-independent Task-Control Management library (TCM). The translated code can then be compiled using standard C++ compilers and 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.

4. Doing the Challenge Task

As mentioned before, the Challenge is to have an autonomous mobile robot attend the National Conference on Artificial Intelligence. More specifically, the Challenge rules (www.cs.utexas.edu/users/kuipers/AAAI-robot-challenge.html) are to have the robot perform the following subtasks:

1. Start at the front door of the conference center;
2. Navigate to the registration desk (ideally by locating signs and/or asking people and/or following people – at this point, the robot does not have a map of the building);
3. Register: stand in line if necessary, have the robot identify itself, receive registration material, a map of the conference center, and a room number and time for its talk;
4. Interact with other conference attendees (ideally recognize participants by reading nametags or recognizing faces and schmooze – striking up brief personal conversations);
5. If requested, perform volunteer tasks as time permits, such as "guarding" a room or delivering an object to another room;
6. Get to the conference room on time, using map received in step 3. This may involve riding an escalator or elevator.
7. Make a two-minute presentation about its own technology, and answer questions.

For our first year of the Challenge, we decided to do all of the subtasks except #4 (schmoozing) #5 (volunteer duties), and having the robot itself answer questions from the audience. In addition, the human interaction in #2 was limited to interaction with one of the developers. In future years, we will expand the scope to include all subtasks and enable arbitrary conference participants to interact with the robot.

The next sections describe in more detail the major subsystems for each of the Challenge tasks.

4.1 Getting to the Registration Area

The first part of the Challenge is to have GRACE start at the entrance to the conference center and find the registration area by interacting with a person. We used an off-the-shelf speech recognition system, IBM's ViaVoice, to convert from spoken utterances to text strings. The text strings were then parsed and interpreted using Nautilus, NRL's in-house natural language understanding system,

[Perzanowski, et. al., 2002; Perzanowski, et. al., 2001; Perzanowski, et. al., 1998; Wauchope, 1994]. The output of this component is something like a logical form used in standard predicate logic. This representation is then mapped to a message, or a series of messages, which is then sent to other modules through an IPC interface. The mapping code was written in TDL and it, and the IPC interface, was developed specifically for the Challenge.

To achieve a goal, we interleave linguistic and visual information with direction execution. If there are no directions to be followed, GRACE performs a random walk until a human is detected (for the Challenge this past year, human detection was done using a laser scanner; in future years, we will incorporate both vision-based and stereo-based detection of people). GRACE then engages the human in a conversation to obtain the 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 in order 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. Instead, we developed a Palm Pilot-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," we assume that the command is executed immediately, before execution of any other instructions. 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. Once all the available directions have been executed and successfully completed, GRACE concludes that either she 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 she continues with the next goal.

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 fifty meters," the human's input is mapped to a representation something like the following:

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

4.2 Riding the Elevator

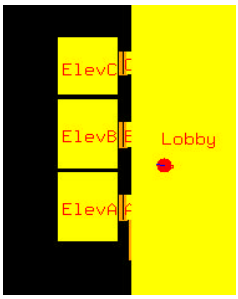
3. The elevator in Edmonton

As mentioned previously, the registration area in the Shaw Convention Centre is not on the same floor as the street entrance. Our choices in addressing this 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 3).

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 facing it. 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 instance, given that the robot is positioned as shown in Figure 4, the system will see laser readings like those in Figure 5.

While people can readily make out the shapes of the elevators in the laser points, having the robot find 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



4. The simulation environment

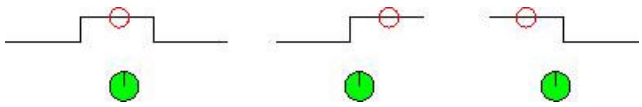
5. Raw laser points

- 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 6). 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. While these 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 of Figure 5, the system detects the two elevators shown in Figure 7.

One difficulty is that some patterns that are not elevators can actually look similar to the patterns in Figure 6. For instance, Figure 8 illustrates two types of patterns that are not elevators. Note that, in practice, some patterns that initially look good (e.g., the two patterns on the right in



6. The three valid elevator patterns

7. The system, fully settled, with two elevators discovered

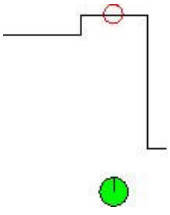
Figure 6) may actually turn out to be bad patterns when more information is acquired (by moving around).

After the robot detects an elevator, it gets into position and waits for the door to open. While 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

8. Two invalid elevator patterns

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.



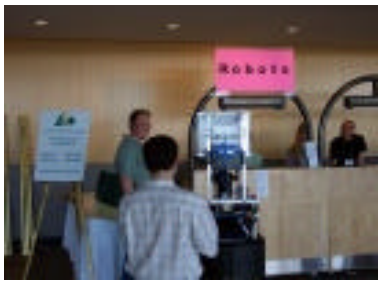
9. The unusual pattern at the Challenge

While 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, were made primarily of laser-invisible glass (see Figure 1). To solve this problem, we discreetly put a single strip of stylish green tape all around the area, just at laser height. This neatly solved the problem and drew little attention from 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 (see Figure 9). 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 above, 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 sensor, based on an electronic altimeter, to determine which floor the robot is on.

4.3 Finding the Registration Booth



10. The Robot Registration Sign

Once GRACE reached the registration area (Section 4.1), the next task was to move up to the registration desk. This 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 10).

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 and with differing priorities, while maintaining 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 up to six different modes, including the PTZ_SET and LOOK_AT modes that were used with GRACE. The PTZ_SET mode allows software external to 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 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.

Due to the configuration of the registration area at the Shaw Centre, GRACE was approximately 15-20 meters from the registration desk when she 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 15-20 meters, with the camera's zoom set to the widest angle (45 degree field-of-view), the sign was only a few pixels in size and nearly impossible for SVM's blob detection operator to find. In order 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, due to 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, 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 towards 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; these were then translated into global coordinates by the TDL code, which provided both sign and robot location updates to SVM, as well as corrected 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 so as to maintain the entire sign within the image, thereby reducing the chance of losing the sign and producing more accurate estimates of the its location. This part of the task was considered completed when GRACE reached a distance of two meters from the desk.

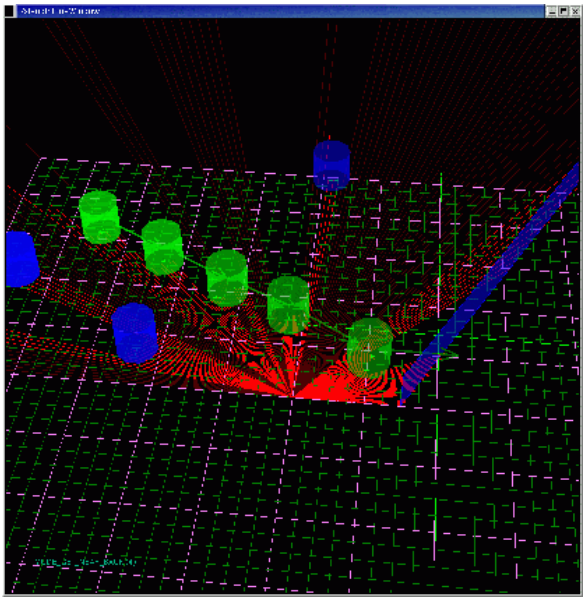
4.4 Standing in Line

Once GRACE was near the registration desk, she proceeded to register. First, however, she 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 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, while maintaining a minimum socially acceptable separation distance. GRACE also uses this information to ensure that once in line she 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 & 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: those that might be people, those that are likely walls, and other (Figure 11). 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 ~50cm) 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 to be a wall. Occlusions in the range data (as seen in Figure 11) are compensated for by comparing wall clusters to one another to determine if a single wall segment can explain them. If this is the case, then those clusters are combined to provide a better estimate of the orientation and location of the walls.



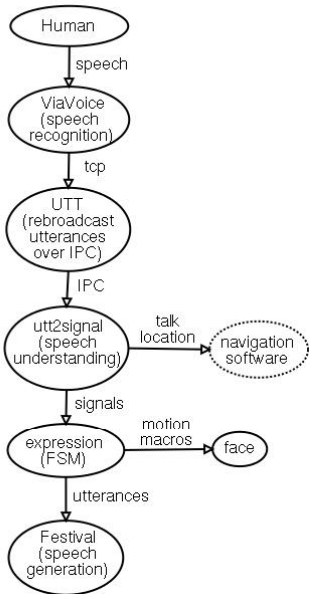
11. GRACE's perception of people in line

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 considered to be not in line. Once the line is found, GRACE moves towards 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 her, maintaining the personal space between the robot and that person. Once near the registration desk, GRACE maintains a stand-off distance until the person in front leaves. When there are no more people in front of GRACE, she drives to a set distance from the registration desk and then begins to register.

4.5 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 the location and time of her talk.

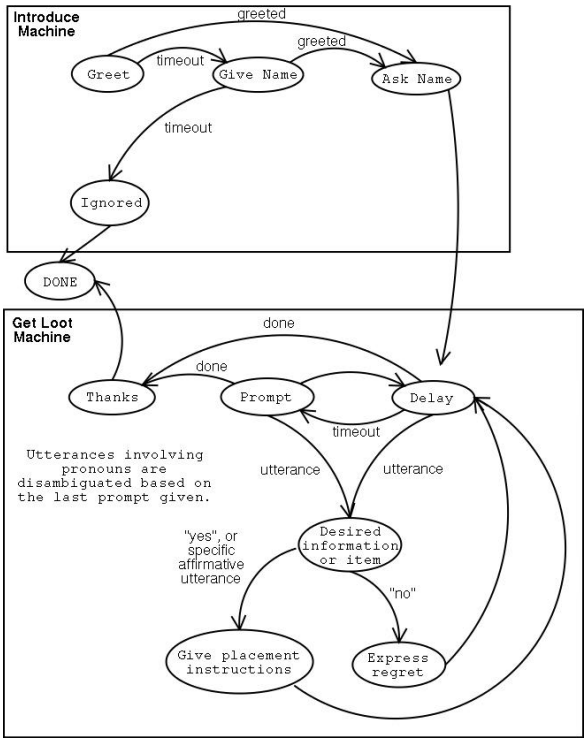
Figure 12 illustrates the data and control flow for a typical interaction cycle with the robot. A wireless microphone headset is used to acquire speech, which is then converted to text by ViaVoice. ViaVoice has the ability to read in a user-specified BNF-style grammar, which it then uses to assist in 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. Since the breadth of interaction involved in performing the registration task is rather limited, we were able to achieve satisfactorily accurate recognition.



12. Information flow for the registration task

ViaVoice transmits the utterances that it recognizes as strings over TCP in its own proprietary format. 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 below). In addition, utt2signal is responsible for dispatching any raw information gleaned from the utterances to the appropriate process. For instance, if the registrar tells GRACE the location of her talk, utt2signal informs the navigation software of this.

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 consist of basic face movements, utterances, non-face primitives (such as pauses), and other macros. Even more



13. Simplified FSM for the registration task

powerful is the ability to create and execute hierarchical finite state machines (see Figure 13). The FSMs can execute actions when entering a state and can transition based on signals received from other processes (e.g., utt2signal – hence the name). Figure 14 shows a small sample of the script used for the registration task.

Since 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 finite state machines. In the future, this will allow much more complex interactions to be driven without worrying about computational requirements.

GRACE's face (Figure 15) is one of the most important aspects of her ability to interact with humans. It is used for both emotional expression and for simple gestures, since GRACE lacks any conventional manipulators. The face is based on an implementation of the simple face in [Parke & Waters, 1996]. It incorporates a muscle-level model of face movement to allow semi-realistic 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.

```
# 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
```

14. Sample expressions and FSM's for the registration task

15. Grace's Face

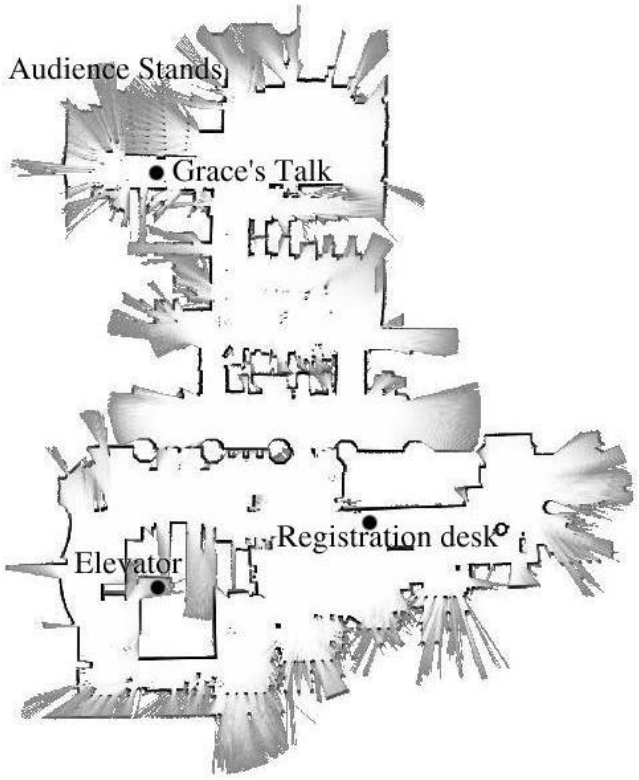
Last, but not least, is GRACE's ability to generate speech. We use a version of Festival that was modified to enable it to generate phonemes for a given utterance, which are then processed to extract lip-synching information. Festival performed admirably, overall, with two notable exceptions: it tends to speak in a monotone and cannot handle acronyms. While it is possible to embed pitch changes in strings sent to Festival, this 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. Since an utterance of just numbers was parsed as a statement of the time of GRACE's talk, this could cause some confusion. However, GRACE was able to recover from such mistakes, due to 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 she gave (i.e., what state of the FSM she 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 non-specific statement was about the new prompt, even if she has only said a syllable or two of it. This obviously caused some problems, as the potential existed for her belief of the state of the world to get out of sync with reality, resulting in very unnatural interaction.

4.6 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 she had built previously and was saved on disk. The map was used to help GRACE make her way from the registration desk to the talk venue. The map-based navigation task was comprised of three main technologies: map-building, localization, and navigation control.



16. Map built by GRACE of Shaw Convention Centre

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. This data was then used to build a map through a process called *scan matching* [Lu & Milios, 1997]. The implementation of our scan-matching algorithm was adapted from a software package provided by Dirk Hahnel at the University of Freiburg [Hahnel 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 16. This map is 89.4 x 10.8 m, with a resolution of 10cm/grid cell. The black pixels represent regions of space with a high probability of occupancy, such as walls, chairs, etc. Similarly, the white areas are regions of space with a low probability of occupancy. Not shown in this image are regions of space where no data could be collected (i.e., 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:

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

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 & 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 negative reward to poses close to obstacles. The planner uses value iteration to assign a value to each cell; this value 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. This allows the planner to compensate for small errors in localization and changes to the environment that could invalidate certain paths.

4.7 Giving the Talk

Once GRACE navigated to the lecture area (in the Exhibition Hall), she gave a talk about the technologies that comprised her. 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 those 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. Inference rules in the system are compiled into a feed-forward logic network. This gives it *circuit semantics*: the inputs of the network monitor the truth-values of premises as generated by the sensory systems and the outputs 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 sounds inefficient, the rule engine can run a base of 1000 Horn rules with 10 conjuncts each, updating at 100Hz (100 complete reevaluations of the knowledge base per second), using less that 1% of the CPU. Using a generalization of deictic representation called *role passing*, the network is able to implement a limited form of quantified inference – a problem for previous behavior-based systems. Rules may 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: a marker-passing semantic network provides a simple mechanism for long-term declarative memory, while role passing allows variables within inference rules to be bound to behaviors and signals within the system. The former allows the system to answer questions about its own capabilities, while the latter 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 (e.g., 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 myself.
- If `activity` deactivates itself, also deactivate myself.

Since 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 finite-state machines, 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, while statically compiling programs to a network of parallel finite-state machines.

17. GRACE gives a talk

To give a talk (Figure 17), 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 SIMD 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. She then discusses the topic by continually selecting and explaining the "highest priority" relevant, unexplained, node. Priorities are computed off line using a topological sort so that if topic A is required to understand topic B, A will always have higher priority.

By continually reselecting the highest priority relevant, unexplained node using circuit semantics, the system can respond instantly to changes in relevance when, for example, an unexpected contingency during a demonstration opens up an opportunity to explain a feature. 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, the system can determine whether it has moved locally up, down, or laterally in the hierarchy, or whether it has made a non-local 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.

5. Discussion and Summary

On Wednesday July 31, GRACE attempted the AAAI Robot Challenge, in front of hundreds of interested onlookers and the media. GRACE successfully completed each of the subtasks described above, with a minimal amount of 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 compares to about 20 minutes taken by the other entry that attempted the complete Challenge – the CoWorker built by iRobot – but that robot was remotely teleoperated by a person in the convention center.

While each of the subtasks was successful, and GRACE successfully completed an end-to-end run, each subtask also demonstrated need for improvement. Probably the most critical problem was based on our use of ViaVoice for speech recognition. ViaVoice has troubles with background noise and stress in the speaker's voice. It can recognize only grammatically correct sentences, and returns only the best parse, whereas we would like to get back several of the most probable parses and use speech understanding to determine which is most likely given the context. To try and remedy this, we are in the process of switching to Sphinx for speech recognition (see http://www.speech.cs.cmu.edu/sphinx). This 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. This would enhance GRACE's appearance as an independent entity and enable random interaction.

While the human-robot interaction (aside from the speech recognition) worked relatively well, there were areas for improvement. For instance, gesture recognition, which works on the NRL robots, was not successfully integrated in time for GRACE. In addition, NRL has developed an ability to talk about semantic entities in the environment (e.g., "turn down the next corridor"), but the ability to recognize these features is not yet integrated on 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, in order to give it time to enter and exit before the doors closed. Part of this was due to the fact that the robot did not recognize changes to the environment fast enough. Also, the robot also did

not have any way of determining which floor it was on (we are working on this by developing an electronic altimeter – see Section 4.2).

Visual servoing to the registration desk suffered from several problems. First, as described in Section 4.3, changes in lighting could cause the recognition algorithm to fail, and so the system had to be retrained on a periodic basis. Second, when the robot was far away, the sign appeared too small to be readily identified; but, zooming in gave a very small field of view, which slowed the search for the sign considerably. To deal with this, we are considering a multi-scale 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. This 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 had problems with ViaVoice, as described above. 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, this 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 – this 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, nametag reading, and face recognition. We plan to incorporate capabilities for the robot to "schmooze" with other participants and to 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 trying to attend the conference.

References

[Burgard et. al., 1998] W. Burgard, A.B. Cremers, D. Fox, D. Hahnel, G. Lakemeyer, D .Schulz, W. Steiner, and S. Thrun. "The Interactive Museum Tour-Guide Robot." In *Proceedings of the AAAI Fifteenth National Conference on Artificial Intelligence*, 1998.

[Hahnel et. al., 2002] D. Hahnel, D. Schulz, and W. Burgard. "Map Building with Mobile Robots in Populated Environments." In *Proceedings of Conference on Intelligent Robotics and Systems*, 2002.

[Ko & Simmons, 1998] N.Y. Ko and R. Simmons, "The Lane-Curvature Method for Local Obstacle Avoidance." In *Proceedings of Conference on Intelligent Robotics and Systems*, Vancouver, Canada, 1998.

[Konolige, 2000] K. Konolige. "A Gradient Method for Realtime Robot Control." In *Proceedings of Conference on Intelligent Robotic Systems*, 2000.

[Lu & Milios, 1997] F. Lu and E. Milios. "Globally Consistent Range Scan Alignment for Environment Mapping." *Autonomous Robots*, **4:333-349**, 1997.

[Maxwell et. al., 2002] B.A. Maxwell, N. Fairfield, N. Johnson, P. Malla, P. Dickson, S. Kim, S. Wojtkowski, T. Stepleton. "A Real-Time Vision Module for Interactive Perceptual Agents." *Machine Vision and Applications*, to appear 2002.

[Nakauchi & Simmons, 2002] Y. Nakauchi and R. Simmons. "A Social Robot that Stands in Line." *Autonomous Robots*, **12:3** pp.313-324, May 2002.

[Parke & Waters, 1996] F. Parke and K. Waters. *Computer Facial Animation.* A.K. Peters, Ltd., December 1996, ISBN 1-56881-014-8.

[Perzanowski et. al., 1998] D. Perzanowski, A.C. Schultz, and W. Adams. "Integrating Natural Language and Gesture in a Robotics Domain." In *Proceedings of the International Symposium on Intelligent Control*, IEEE: Piscataway, NJ, pp. 247-252, 1998. [Simmons & Apfelbaum, 1998] R. Simmons and D. Apfelbaum. "A Task Description Language for Robot Control." In *Proceedings of Conference on Intelligent Robotics and Systems*, Vancouver, Canada, 1998.

[Perzanowski et. al., 2001] D. Perzanowski, A.C. Schultz, W. Adams, E. Marsh, and M. Bugajska. "Building a Multimodal Human-Robot Interface." In *IEEE Intelligent Systems*, IEEE: Piscataway, NJ, pp. 16-21, 2001.

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

[Thrun et. al., 2000] S. Thrun, D. Fox, W. Burgard and F. Dellaert. "Robust Monte Carlo Localization for Mobile Robots." *Artificial Intelligence*, **101:99-141**, 2000.

[Wauchope, 1994] K. Wauchope. *Eucalyptus: Integrating Natural Language Input with a Graphical User Interface*. Tech. Report NRL/FR/5510-94-9711, Naval Research Laboratory: Washington, DC, 1994.