GRACE and GEORGE:
Autonomous Robots for the AAAI Robot Challenge

Reid Simmons, Allison Bruce, Dani Goldberg, Adam Goode, Michael Montemerlo, Nicholas Roy, Brennan Sellner, Chris Urmson

Carnegie Mellon University
{reids+challenge@cs.cmu.edu}

Alan Schultz, William Adams, Magda Bugajska, Matt MacMahon, Jessica Mink, Dennis Perzanowski, Stephanie Rosenthal, Scott Thomas

Naval Research Laboratory

Ian Horswill, Robert Zubek

Northwestern University

David Kortenkamp, Bryn Wolfe, Tod Milam

Metrica, Inc.

Bruce Maxwell

Swarthmore College

Abstract

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 on a pair of robots named GRACE and GEORGE. This paper describes the second year effort by the GRACE team, the various techniques each participant brought to GRACE, and the integration effort itself.

1. Introduction

The main objectives of the AAAI Robot 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 task 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 for a robot to attend the AAAI National Conference on Artificial Intelligence as a participant – the robot must find the registration booth and register, interacting with people as needed, then, with a map from the registration packet in hand-analogue, find its way to a specified location in time to give a technical talk on itself. 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. However, compromises will be necessary in order to allow 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 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. While research has been done in 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. The integration of these technologies is a major challenge.

In addition, the members of the GRACE team believe that the type of collaborative work that is needed to achieve the necessary integration will help advance robotics. We realize that integrating hardware and software from five institutions is 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. This year, we extended our architecture and infrastructure from our first effort, and added additional capabilities to the system.

Figure 1. GRACE (left) and GEORGE (right).

2. Robot Hardware

GRACE (Graduate Robot Attending ConferencE) is built on top of an iRobot B21 mobile robot, while GEORGE (GraduatE RObot attendinG conferencE) is built on iRobot's B21R. Both robots have an expressive computer-animated face displayed on a flat-panel LCD screen, as well as a large array of sensors (see Figure 1). The sensors that come standard with the B21 and B21R include touch, infrared, and sonar sensors. Near the base is a SICK scanning laser range finder that provides a 180-degree field of view.

Both robots have 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. The robots speak using high-quality speech generation software (Festival), and receive human responses using a wireless microphone headset (a Shure TC Computer Wireless transmitter/receiver pair).

All the software runs on board. Two 500 MHz systems on GRACE and two dual P III 800 MHz systems on GEORGE, running Linux, run most of the autonomy software. An additional small laptop on each robot, running Windows, runs the speech recognition and natural language speech-to-text software. Note that *only* the speech recognition software runs on the laptop – all speech understanding software is executed on the main robot computers. In addition, there is a separate processor for the Metrica stereo head. Each robot is equipped with a Linksys wireless access point to connect the robot to the outside world (for debugging, monitoring, and backup).

3. Software Architecture

Following the design decision from the previous year, the system was designed as a set of independent programs that communicated via message passing. This facilitated distributed development and simplified testing and debugging. We continued to use the IPC package (www.cs.cmu.edu/~IPC) for (nearly all) communications, due to its expressiveness, ease of use, and familiarity.

Most of the software was written in C or C++ (using the GCC 2.96 compiler), running under Red Hat Linux 7.2, 7.1, and 6.2. Exceptions included the use of a Windows laptop to run ViaVoice (www.ibm.com/software/speech) and the use of Allegro Common Lisp for NRL's Nautilus natural language understanding system. In addition, OpenGL, Perl and Festival (www.cstr.ed.ac.uk/projects/festival) were used for the computer-animated face and speech generation.

The computer-animated face, the coordinating Poobah process (Section 4.7) 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 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.

Finally, we used Microraptor (http://gs295.sp.cs.cmu.edu/brennan/mraptor/) to provide seamless cross-machine process control. The Microraptor system consists of a daemon running on each machine and any number of clients (usually running on wirelessly-connected laptops). The daemons share a single global configuration file and provide remote process spawning, killing, and monitoring capabilities. Using the Microraptor TDL library, the Poobah (Section 4.7) was able to start and stop processes quickly and easily. In addition, the system provided a simple way for multiple users to view the output of and interact with the plethora (30+) of processes that make up the GRACE system. Please note that Microraptor is still under heavy development, has not been released beyond a few trial projects, and that the authors cannot at this time provide any support whatsoever to anyone desiring to use it. Any interested parties should contact the authors via the above website to receive notification upon official release.

In addition to continuing the effort to integrate the vast amount of software that had been developed by the participating institutions, this year we also decided to duplicate the entire system onto a second, separate hardware platform (Section 2).

4. Performing the Challenge Task

As mentioned above, the Challenge is to have an autonomous mobile robot attend the National Conference on Artificial Intelligence. More specifically, the robot is to 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.

This year, we planned to do all of the subtasks, but in the end had to do without the abilities for schmoozing and volunteering. The robots accomplished a new subtask this year by answering simple questions from the audience by themselves. In addition, just like last year, the human interaction on the way to the registration desk 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 sections describe in more detail the major subsystems for each of the Challenge tasks.

4.1 Getting to the Registration Area (GRA)

The goal of this portion of the challenge is to find the registration area by interacting with people. 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., 1998; Wauchope, 1994]. Nautilius’ output, in a format similar to that used in standard predicate logic, was mapped to IPC messages that represented the command or information given. To achieve a goal, we interleave linguistic and visual information with direction execution (see Figure 2). If there are no directions to be followed, the robot performs a random walk until a human is detected. The robot then engages the human in a conversation to obtain directions to the destination in question. The task is completed once the destination is reached, as determined by an explicit human confirmation or by the robot visually detecting the goal (Section 4.2). Execution monitors run concurrently throughout the task 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.

The directions can consist of four types of utterances:

- simple low-level movements,
- mid-level movements requiring more perception,
- new sub-goals, and
- commands relative to named objects.

Simple low-level commands, such as "turn left" and "go forward five meters," provide the most direct control of the robot. Higher level instructions, such as "go down the hallway on your right" use more of the robots' perception

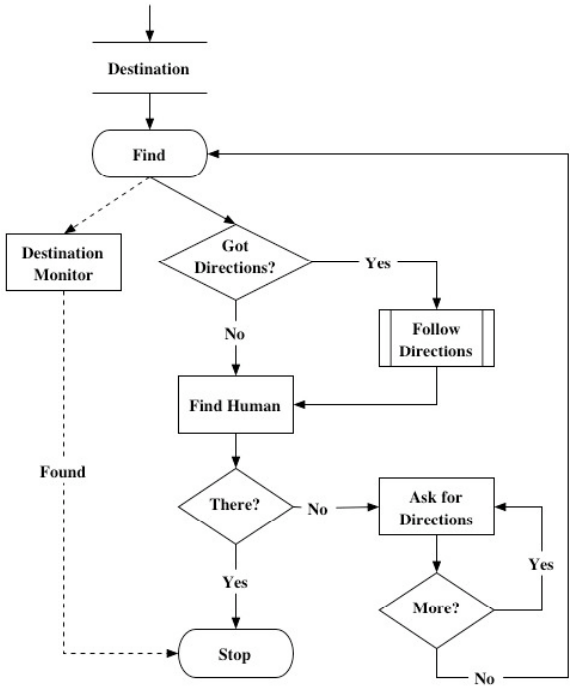

Figure 2. Direction Taking

and rely less on the direction-giver to accurately estimate distances and angles. An utterance like "Go to Cholula Hall" gives the robot a more immediate sub-goal to accomplish its current goal. Finally, one can name objects and refer to locations relative to them: "That is a pillar [pointing]. Go behind the pillar." In addition, the robots can ask simple, yes-no questions such as "Am I at the registration desk?" and "Can you help me?".

Places (destinations) are organized topologically, including containment relationships and route connectivity. Every obtained direction is part of a memorized route between two, possibly unspecified, places. For each route, directions are executed sequentially. For directions specifying an intermediate destination, such as "take the elevator to the second floor," an intermediate goal is instantiated (getting to the elevator), and the logic is recursively applied to the new goal.

Once all the available directions to the current goal have been executed, the robot concludes that either it has arrived at the destination or additional information is required to reach the goal. If the robot perceives the destination before all the directions are executed, the remaining ones are abandoned, and it continues with the next goal. Thus, suppose the robot asks a human bystander "Excuse me, where is the registration desk?" and the human responds, "Grace, go over there <accompanied by a gesture>. Take the elevator to the ground floor. Turn right. There is a pillar in front of you. Go behind the pillar." The directions are mapped to a representation something like:

```
Find Registration Desk:
    Find Elevator ('ground floor');
        Go over there <gesture direction>;
    Turn right;
    Name Object(Front,'pillar').
    Go Relative('pillar', behind).
```

The use of spatial relationships is a new capability this year. The robot can refer to locations relative to named objects for environment descriptions e.g., "There is a desk in front of me and a doorway behind it," and issued directives, e.g., "Go around the desk and behind the pillar." In addition, users can inquire about the robot's environment and label objects. For example, a user may ask the robot, "What do you see?" and the robot could respond, "There are objects in front of me and on my left." Spatial relationships are derived from an evidence grid map that accumulates occupancy information from multiple sensors. This short-term map is then filtered, processed, and segmented into environment objects. The spatial reasoning code then generates a high-level linguistic description of the overall environment and a detailed description for each object. Objects are clusters of occupied cells in the map that can be assigned labels by a user. These capabilities establish a more natural communication mechanism between people and robots, especially for novice users [Skubic, et al., 2002].

4.2 Finding the Registration Desk (FRD)

Once the robot reached the registration area (Section 4.1), the next subtask was to move up to the registration desk. This involved three related sub-subtasks: (1) searching for and visually acquiring the sign indicating the registration desk; (2) confirming that the correct sign had been located by reading the text printed on it; and (3) approaching the desk in a manner such that the standing in line module could detect the line. This year, we narrowed the definition of this module to (1) passively (i.e. with no module-directed camera or robot movement) detecting pink blobs and (2) verifying that the detected pink blob is the correct sign by reading any text on it. Thus, this module did not have to deal directly with any locomotion issues, instead allowing the Poobah (Section 4.7) to handle them. This reassignment of responsibilities was necessary to facilitate a smooth handoff between this module and the standing in line module. Since the registration area itself was unreachable by the robots (no elevators or ramps were available), we used our own (approximately 0.5 x 1.0 meter) large pink signs – one with the text “Humans” and one with “Large Robots” – on the NRL exhibition booth.

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 and GEORGE.

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 mode that was used in this project. 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. The software for finding the registration desk was written using TDL.

The convention center layout was such that the registration desk was rather close (roughly 10 meters) when the robot first came into eyeshot of it. This, combined with an improved version of SVM and the module code, allowed us to use the camera to passively search for pink blobs at its widest field of view (45 degrees) while the robot was moving. This behavior was initiated during the Get to Registration Area (GRA) (Section 4.1) module's execution, when the robot was told that it was in the general vicinity of the registration area by the human giving it directions. We avoided running the blob detector at all times in order to minimize false sign detections (a number of items used by the maintenance crews were identical in color to our signs).

As soon as a pink blob was detected, the Poobah (Section 4.7) was notified and immediately paused the GRA module. The Finding the Registration Desk (FRD) module then proceeded to zoom in on the sign in a number of stages (to allow for realignment with the sign) and read any text on the sign. This text (or lack thereof) was then compared with a hard-coded target string (in this case, "Large Robots"). If the strings matched, the Poobah shut down the GRA module and began the approach to the desk. If they didn't match, control was returned to the GRA module and our passive blob detection continued. In testing and during the run, we saw a very low number of false positive blob detections while detecting all visible signs. If the GRA module ran to completion and control was handed to the FRD module, a search was performed by scanning the world with the camera's widest possible field of view (once a blob was detected, execution proceeded as before).

Once the registration sign was found and verified to be the correct sign, we calculated the rough position of the registration desk based on the blob elevation measure provided by SVM and the robot's odometry. At this point, the Poobah (Section 4.7) moved the robot to a position two meters in front of and two meters to the side of the desk. Before the move, the Poobah (Section 4.7) also enabled the Stand In Line (SIL) (Section 4.3) module. The goal was to move off to the side of any potential line in order to give the SIL module a clear view of the line. During the move, the SIL module continuously attempted to detect a line. As soon as it did so, the Poobah shut down FRD and ceded control to SIL. If SIL had not detected the line by the time the robot finished its move, the Poobah assumed no line existed, moved directly to the registration desk, and proceeded to register (Section 4.4).

During this year's run, the FRD module interrupted the GRA module upon detection of the "Human" sign. It then proceeded to zoom in on the sign, failed to read the text, and concluded that it was the wrong sign. In the interim, the human giving directions indicated that the robot was at the registration area, which terminated the GRA module and handed control to the FRD module. A search commenced, during which the correct ("Large Robots") sign was detected and verified. Unfortunately, the SIL module failed to detect the line, and the Poobah ended up driving GRACE directly to the head of the line.

4.3 Standing in Line (SIL)

Once the robot was near the registration desk, it proceeded to register. First, however, it attempted to wait in line, like any polite conference attendee. GRACE and GEORGE use a combination of an understanding of personal space and range information to stand in line. They use 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 and GEORGE also use this information to ensure that once in line they do 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].

The robot 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 3). 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 this limited context.

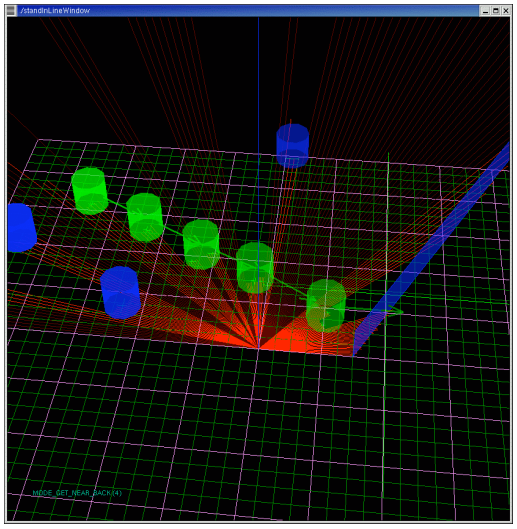

Figure 3. GRACE's perception of people in line

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 3) 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.

This year's version of the Stand In Line (SIL) module was much more robust to the robot's initial position than the 2002 version, at least during testing (it failed to detect the line during the actual run). Roughly the same algorithm was used, but the assumption that the desk was initially in front of the robot was removed. Instead, the SIL module can be run in either a monitoring mode, where it reports the location of the head of the line it detects (if any), or in an active mode where it takes control of the robot and actually stands in line. This allows the Poobah (Section 4.7) to move the robot about according to the information reported by the FRD module and then hand off control to the SIL module as soon as it detects a line whose head is close enough (within one meter) to the FRD module's estimate of the location of the registration desk.

The "stand in line" algorithm takes a head-of-the-line hint (which is the estimated location of the desk produced by the FRD module) and tries to find the closest wall to that hint. Once the closest wall has been found, the robot searches for the person closest to this wall. 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, the robot moves directly towards the back of the line, intermittently checking for more people in line. When it reaches the back of the line, it moves to a position behind the last person. At this point, the robot only considers the person immediately in front of it, maintaining the personal space between the robot and that person. Upon nearing the registration desk, it maintains a stand-off distance until the person in front leaves. When there are no more people in front of the robot, it drives to a set distance from the registration desk and control passes to the Register module.

4.4 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 and GEORGE at least somewhat as they would with a human. The specific task was for the robot to obtain all the various registration paraphernalia (bag, badge, and proceedings), as well as the location and time of her talk. The implementation of this subtask was identical to the 2002 effort.

Figure 4 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

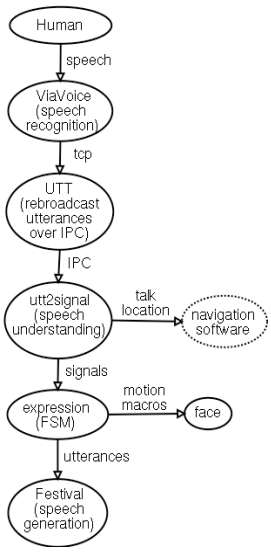

Figure 4. Information flow for the registration desk task

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.

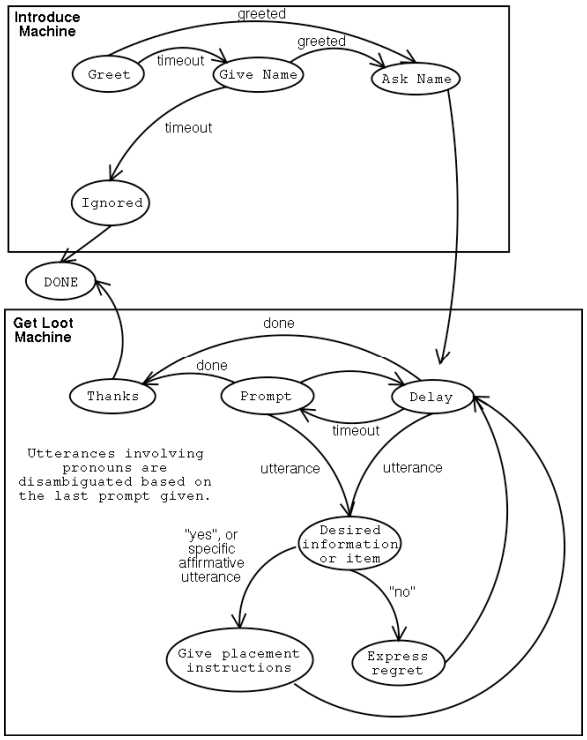

Figure 5. Simplified FSM for the registration task

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. The text strings are then parsed by the utt2signal process. utt2signal performs the same basic function as Nautilus, but is significantly simpler and more 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 the robot the location of its 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 powerful is the ability to create and execute hierarchical finite state machines (see Figure 5). 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).

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.

The robots' faces (Figure 6) are one of the most important aspects of their ability to interact with people. It is used for both emotional expression and for simple gestures, since GRACE lacks any conventional manipulators. GEORGE obviously uses a different face model (Figure 6), but the underlying software is identical. 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.

Figure 6. GRACE's (left) and GEORGE's (right) Faces

New this year was a mood server, which attempts to make the face more dynamic by causing its expression to change based on what is happening to the robot at the time. One shortcoming of last year's system was that the face remained expressionless except during explicit utterances, which often included embedded facial expressions. The

mood server was designed to make the expressions change in a coherent and realistic way by maintaining an internal "emotional state" for the robot. This emotional state and the way that it is updated are based on the Ortony, Clore, and Collins model of the cognitive structure of emotions [Ortony, 1988]. The mood server accepts as input events from the various modules corresponding to interactions with the real world. For example, an event was fired whenever the robot received a desired object during registration. These events were interpreted as affecting one or more of a number of mood axes corresponding to basic emotions, and the underlying estimation of the robot's mood was appropriately updated. This estimation of mood was then used to continuously modify the expression of the face, both during utterances and when the face would have been otherwise static.

Last, but not least, is the system'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, 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 the robot'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. The system disambiguates such statements as "here you go," "no," or "you have it" based on the latest prompt that was given (i.e., what state of the FSM the robot is currently in). However, if the robot prompted the registrar and the registrar began to respond, but ViaVoice did not complete recognizing the utterance until after the system had timed out and begun the next prompt, the robot would believe that a non-specific statement was about the new prompt, even if only a syllable or two of it had been uttered. This obviously caused some problems, as the potential existed for the robot's belief of the state of the world to get out of sync with reality, resulting in a potentially very unnatural interaction.

4.5 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 and GEORGE, however, use a map that they had built previously and saved on disk. The map was used to help the robots make their 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.

The evening prior to the Challenge event, the robot 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 7. This map is 103.1 x 56.8 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).

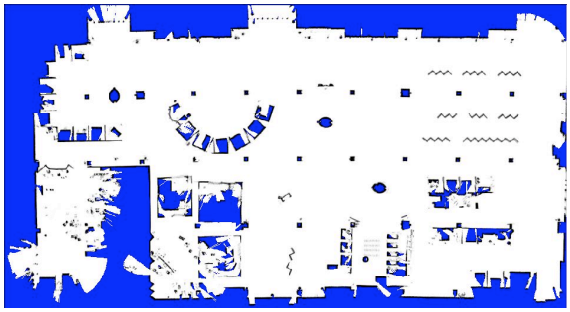

Figure 7. Map built by GRACE of the Acapulco Convention Center

The robots use 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]. The robot is initialized with an approximate starting position, and the distribution of

particles evolves to reflect the certainty of the localizer's position estimate.

As the robot moves, the probability distribution is updated according to:

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

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:

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

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.

The only major difference in this year's implementation was the inclusion of a fiberoptic gyro on GRACE, which yielded much more accurate rotational pose, greatly increasing the accuracy of the generated map.

4.6 Giving the Talk

Once the robot navigated to the lecture area (in the Exhibition Hall), it gave a talk about the technologies that comprised her. The 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.

To give a talk, GRACE and GEORGE use 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, the system 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 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 re-planning. 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.

4.7 The Poobah

One important aspect of overall integration that was lacking in our effort last year was a coordinating process. The various modules were manually serialized and very self-contained; little to no high-level failure recovery was performed, and due to this structure each task was extremely specific to the Challenge. The Poobah is our first effort to address these issues.

The Poobah is written entirely in TDL, and contains one TDL task tree corresponding to each of the modules. It is responsible for launching and killing the appropriate processes at the right times (via Microraptor; see Section 3), coordinating all data flow between modules, and handling all aspects of the overall Challenge task that do not fall within the purview of one of the above-described modules. This affords us a number of advantages:

1) The modules are insulated to as high a degree as possible from Challenge-specific details. This allows existing work to be integrated much more quickly by keeping the modules generalized.

2) If a module fails, we can fall back to a previous module or skip ahead, depending on the context and reported details of the failure.

3) Data can be easily maintained across multiple modules, allowing non-serial data flow among modules.

4) The majority of the Challenge-specific code can be concentrated in one central location.

In the current implementation, the majority of the tasks are merely serialized by the Poobah. A notable exception is the FRD => SIL => Registration handoff. As detailed in Sections 4.2 and 4.3 above, the Poobah is responsible for passing hints from the FRD module to the SIL module, as well as executing a number of robot moves (and a smattering of utterances) in order to bring the robot into a state in which the SIL module can take over control. Although the Poobah was not used on GEORGE (since we were unable to get Microraptor working), it performed flawlessly on GRACE.

5. Discussion and Summary

On Thursday, 14 August, GRACE and GEORGE attempted the AAAI Robot Challenge, in front of interested onlookers. While some portions of the software performed quite well – even better in some cases than last year – other parts did not. We also experienced a few hardware problems that confounded our efforts. Overall, almost all errors could be traced to integration problems and not to the underlying algorithms.

Although both robots are similar and GEORGE is a much newer robot, one of GEORGE's computers had an older version of Linux installed, a requirement imposed by the manufacturers' underlying control and interface code. Specifically, GEORGE's two machines run RedHat 6.2 (2.2 kernel) and 7.3 (2.4 kernel), as opposed to RedHat 7.2 (2.2 and 2.4 kernel) on both of GRACE's machines. We believe this might be the root cause for several yet to be diagnosed problems.

Possibly related to a difference in operating systems, we could not get the Microraptor system (and therefore the Poobah) running properly on GEORGE, and thus could not automatically start and monitor the 30+ processes that were required. This required us to hand start and sequence the processes, something that we learned last year was difficult at best and prone to errors.

While we wanted both robots to run in quick succession for the whole challenge, we decided that GEORGE's software was not ready and were going to have GEORGE just give the talk, with him waiting in the talk area for GRACE to arrive. We took GRACE for her run. Unfortunately, at the start of the run, a power drain from a damaged PC-104 stack browned out the SICK laser. This, in combination with the earlier (known) failure of the sonars due to humidity, deprived GRACE of all active sensors, and she promptly ran into a coffee table. GRACE then sat, unresponsive, while the power problem was diagnosed. We decided to see what GEORGE could accomplish, with us manually starting the code.

GEORGE had some new code that was not on GRACE, which would allow us to demonstrate the ability to process high-level spatial commands, such as, "go to the other side of the table" or "go between the table and the chair," however this code was not fully integrated into the system. Although not diagnosed, we believe that this code was either making too many requests of the IPC system or causing the system to wait for responses from it, which along with the human giving a lot of requests in succession, caused the natural understanding system to lag far behind the rest of the system. (GRACE did not have the spatial referencing system, and did not suffer from this problem.) This lag time made speech and gestures difficult to use, and we resorted to moving GEORGE to the talk area to get him ready to give the talk.

The new emotional model that drove the facial expressions (and in some cases the behavior) of GRACE and GEORGE performed well. In fact, as GEORGE suffered from an inability to receive valid directions, as described above, GEORGE's face became more and more panic stricken! This was a programmed response to not receiving directions and progressing to the goal. By the end of the initial phase of the run, GEORGE appeared to be as horrified as his handlers.

Meanwhile, GRACE's power problem was diagnosed and repaired. Further, after several attempts, we decided to not run the speech recognition system (an off the shelf product), which was having problems understanding the human, and instead, typed the English language commands directly to the natural language understanding system. This worked well, and we were able to get GRACE through the initial task of getting near the registration desk.

GRACE did a great job of recognizing and reading the signs indicating the place for robots to register. She found the pink of the "Humans" sign and did not initially correctly read the text, found the pink of the "Robots" sign, correctly read the sign and passed control to the code to stand in line (Sections 4.2, 4.3, 4.7). Here, GRACE again had trouble seeing the line and proceeded to move to the front of the line.

The interaction with the human at the registration desk went reasonably well (this person was one of the judges), and GRACE was given a map built earlier of the exhibit hall and the place where the talk would occur. GRACE navigated quite well, despite a crowd of people, to the area for the talk.

Once in place, GEORGE proceeded to give the talk. The talk code was written to be generic and understand which robot was giving the talk so the robot could refer to itself and the other robot correctly. The talk, shortened this year, was performed well to a small crowd. To cap off the performance, GEORGE answered questions from the audience, using its knowledge base to match against key words from the questions.

As stated earlier, while some things worked much better this year, we also experienced many problems – however, we believe these errors, as shown above were mostly integration related. So why did we still have integration errors in the second year of this project? The problem was the introduction of a second, not quite similar robot. A large amount of time this year was spent getting a similar environment on the second robot, and on getting all of the software to run. While a source revision control system was in use, a sizable part of the code that was considered external to the actual research code was not under revision control, and yet this code needed to be modified for various reasons related to the differences in platforms, resulting in failures in the research code of the other teams. A lesson learned for next year – we are bringing this code under revision control.

Speech recognition, not a part of the research code, but an off the shelf product, still performed poorly this year. We did methodically evaluate other products for speech recognition, but did not find any with better performance. We are currently working with one of the top speech

understanding research groups and hope to have a better system next year.

What went right? The general direction understanding code worked better this year. The robots could read the signs to determine which was the correct line. The navigation code ran extremely well this year. And finally, the robot not only gave a good talk that was not canned, but also answered questions.

With two operational robots to use for development this year, one at CMU and the other at NRL, our first task, currently in progress, is to make all of the code we expected to run perform properly, with no integration problems. We hope to have this accomplished by the end of October. With integration errors out of the way, and with two robots, we expect to be able to concentrate on research issues.

What is in store for next year? We expect to have GRACE and GEORGE able to understand spatial references. The robots will understand utterances like "go to the other side of the…" and "go around the…." They will be able to do spatial reasoning. Using a recently developed computational cognitive model of perspective taking, the robots should be able to understand directions without errors in the human versus the robots perspective.

An unaccomplished goal for this year, we plan to incorporate capabilities for the robot to "schmooze" with other participants. We would like to have the robot perform its own crowd control. And since we have two robots, we plan to have the robots work together next year.

Acknowledgments

We would like to thank the organizers and the sponsors of the 2003 Robot Competition and Exhibition for this opportunity. In addition, we would like to thank Samuel Blisard (University of Missouri at Columbia) for his help with spatial reasoning software and Jonathan Sabo for help with GEORGE's system administration.

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.

[Ortony, 1988] A. Ortony, G. L. Clore, and A. Collins. *The Cognitive Structure of Emotions*. Cambridge University Press, 1988.

[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.

[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.

[Simmons & Apfelbaum, 1998] R. Simmons and D. Apfelbaum, "A Task Description Language for Robot Control," *Proceedings Conference on Intelligent Robotics and Systems*, October, 1998.

[Skubic, et al., 2002] M. Skubic, D. Perzanowski, W. Adams, and A. Schultz. Using Spatial Language in Human-Robot Dialog. In Proc. of ICRA 2002, Washington, DC, 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.