A Modular Software Architecture for Heterogeneous Robot Tasks

Julie Corder, Oliver Hsu, Andrew Stout, Bruce A. Maxwell
Swarthmore College, 500 College Ave., Swarthmore, PA 19081
maxwell@swarthmore.edu

Abstract

Swarthmore's entries to this year's AAAI Mobile Robot competition won second place in the Urban Search and Rescue event and Third Place in the Robot Host competition. This article describes the features of Frodo and Rose, the two-robot team, and discusses their performance. The most important design feature that made the robots successful was a modular software design and communication interface that allowed for the use of the same fundamental software in three different robot tasks requiring both autonomous and semi-autonomous modes.

1 Introduction

This year Swarthmore competed in the Robot Host and Urban Search and Rescue (USR) events of the American Association for Artificial Intelligence [AAAI] 2002 Mobile Robot Competition. Swarthmore entered a team of two robots, nicknamed Frodo and Rose.

This year's Robot Host event was an expansion of past years' *Hors D'oeuvres Anyone?* event. In addition to serving desserts, this year's competitors also had to serve information to conference-goers in the lobby of the Shaw Conference Center in Edmonton, CA during coffee breaks between sessions. The other change in the event was an explicit emphasis on effective serving and a de-emphasis on human-robot interaction. These changes necessitated changes in our design—which was based on Swarthmore's previous entries—from one emphasizing interaction through conversation (and dashing good looks) to one capable of quickly transitioning between an information-serving module and a snack serving module and effectively covering the requisite area.

The remainder of this paper is organized as follows. Section 2 provides a brief description of the hardware used. Section 3 describes in turn the software used for the host and USR competitions. Frodo and Rose's competition performance is reviewed in Section 4.

2 Robot hardware

Frodo and Rose were a pair of identical Real World Interfaces Magellan Pro robots, with 450 MHz Pentium III computers running Linux interfaced with the motor controls through the serial port. A metal table was mounted on top of each robot base, and attached to that were a Canon VC-C4 pan-tilt-zoom camera and battery-powered speakers. A tray was mounted on the table for serving desserts, and a Happy Hacking keyboard took its place for serving information. While serving information a 5.25" LCD display was also mounted on the table. For the USR competition the table was removed and the camera was mounted directly on the robot base, as the other peripherals were unnecessary for that task.

3 Software description

As in past years, Swarthmore's robots feature a modular design. This allows each aspect of the robots' behavior—speech, vision, and navigation--to be managed independently. In previous years, the modules have communicated through shared memory. While this method was very fast, it introduced a number of synchronization issues that had to be carefully monitored [1,2]. In the new design, all communication between modules is handled through the IPC communication protocol developed at Carnegie Mellon University [3]. Using the IPC protocol permits more independent development of each module. Once a standard set of messages is defined for a module, the structure and behavior of that module can change without affecting any of the other modules.

While the robots are active, each one has a central IPC server running. Each module can subscribe to and send messages to the server on either robot. This allows the two robots to communicate with one another during the serving events. In addition, each module subscribes to a set of common messages, which allows a single command to be sent to initialize, idle, or halt all of the modules simultaneously.

A second change from our previous system is the integration of an event-based model for controlling module actions in addition to a standard state machine approach. Since the IPC communication packets are received asynchronously, each module must react to new commands or information as they arrive. Thus, each module now contains both state-based and event-driven aspects. The event handlers allow Frodo and Rose to respond to sensory input such as people and objects they see in the room while simultaneously moving through the steps of serving using a state machine.

Overall, the module design is as follows. Each process starts by defining the IPC messages it will be sending and to which it will subscribe in order to receive information and commands from other modules. The module then enters a state-based loop. At the beginning of each loop, the process

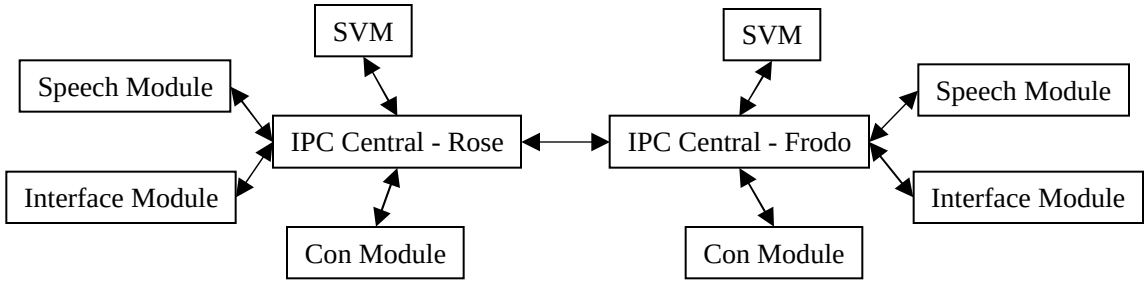

Figure 1: Information flow through IPC. A central IPC server on each robot controls the flow of messages. Each module can submit messages and can listen for messages from other modules.

checks for messages from IPC. If there are messages waiting, the event handler can change the state of the module, send messages to other modules, or take actions as appropriate. The module then executes its main loop based on its current state.

3.1 Robot Host Competition

The modularity of our design was particularly useful in this year's Robot Host Competition. For the two portions of the competition--information serving and dessert serving--we had separate Interface Modules that could be started; all of the other modules functioned identically in both competitions. Since both Interface Modules sent and listened for the same IPC messages, the other modules did not need to know which interface was running.

3.1.1 Con module The Con Module is based on *Mage*, the Magellan control interfaced described in [2], which allows us to control the low-level operation of the robots. The Con module includes both low and high level commands. The low level commands—such as move forward X meters, or turn Y degrees--permit direct control of the robot by another module. The high level commands integrate sensory information with navigation and include: go to a goal point while avoiding obstacles, wander, and track using vision. The high-level actions are built out of multiple behaviors that can be selectively managed to provide different robot behavior. For example, the wander mode can be aggressive, safe, fast, or slow, depending upon which behaviors the calling program invokes.

The control module can receive IPC messages from other modules instructing it to change modes or actions. The only messages it sends out are status messages based on its completion of an action requested by another module; it also informs other modules if a goal-based action times out or completes successfully.

3.1.2 Speech module Unlike the other modules, the Speech module can send IPC messages to modules running on either robot. This allows each robot to initiate a conversation with the other robot.

The speech module has only four states: Idle, Mutter, Converse and Quit. While a robot is serving, its speech module is set to Idle mode. While wandering around the room, the robot can stay in Mutter mode. In Mutter mode, the robot is silent until it is passed the name of a text file. It will then read randomly selected lines from the text file one at a time at a set interval.

In Converse mode, the two robots actually appear to interact with one another. When one robot spots the other robot by detecting the Italian flag that each wears, it sends a message to the other robot requesting a conversation. Depending on the current activity of the second robot, it will either accept or deny the conversation request. If both robots are currently available for a conversation, then the conversation initiator will read the first line from a conversation text file and send an IPC message to the other robot containing the line that the second robot should speak. The second robot speaks its line and then sends an acknowledgement back to the conversation initiator. The conversation ends when the end of the text file is reached or if one of the robots sends an End of Conversation message. This allows a robot to gracefully exit a conversation if someone requests information during the conversation, since serving is always the robots' first priority. For the sake of robustness, a conversation will also end if one robot fails to respond at all. Each robot will only wait for a certain amount of time for a response from the other robot; this ensures that if something happens to one robot, the other will be able to get out of the Converse state and continue serving the guests.

3.1.3 SVM The SVM module (short for Swarthmore Vision Module) provides the link between the camera and all other components of our robot architecture. Conceptually, SVM remains largely as described in [4]; however, the implementation has been upgraded to use IPC for communication with other modules.

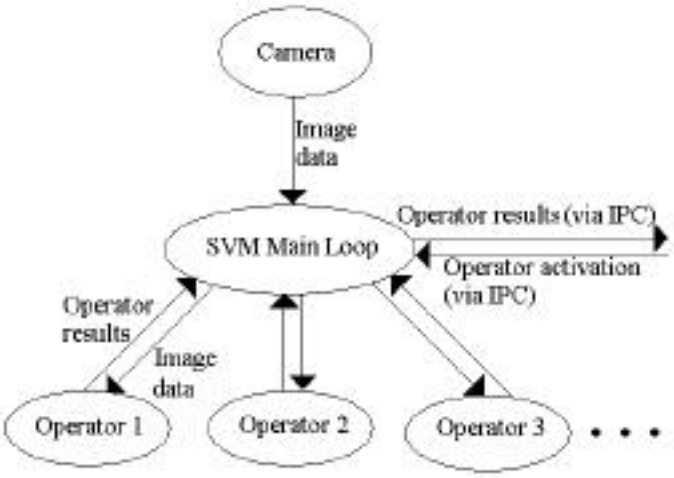

Figure 2: The SVM main loop handles communication with other modules via IPC. It receives image data from the PTZ camera and distributes the data to the appropriate operators, which return the data that is submitted to other modules through IPC.

SVM consists of two main parts: a main loop and a set of image operators. The main loop provides the foundation of the module; once started, it continuously requests new images from the camera. Upon receipt of a new image, it chooses a subset of all currently active operators to be executed and broadcasts results through IPC if necessary. Each operator typically performs one type of analysis on the image, such as face detection or motion detection. Operators can be defined as stochastic or running on fixed timing and can also be dynamically activated and deactivated through IPC messages. When a module requests that an operator be turned on, it can define the location of the PTZ camera or allow the operator to run at whatever location the camera happens to be at.

Several of the vision operators are useful for detecting people. The Pink Blob operator is trained to identify the pink ribbons that robot contest participants wear. The Face operator looks for flesh-colored areas that could be a human face. Once a robot approaches a person, the Motion Detection operator will help to ensure that its goal is, in fact, an animate object (and not a chair or plant). During an interaction with a person, the AAAI Badge Detection operator combines a pattern-based detection of the location of a badge with text recognition to identify a person's name. The Shirt Color operator looks for a solid area of color below a face; the robots can use the shirt information in conversation with conference participants.

In addition, the Italian Flag operator looks for the red-white-green sequence. Since each robot was fitted with an Italian flag during the competition, this operator allowed Frodo and Rose to identify one another. The Tray operator tips the camera down towards the tray holding desserts during the dessert serving competition. Based on the ratio of dark cookies to light tray that it detects, the operator gives the robot information that can be used to determine when the tray is empty so that the robot can return to the refill station. The Hand operator would send a message whenever it detected a hand in front of the camera; this told the robot that someone was reaching for a cookie from the tray, so that the robot could comment accordingly.

3.1.4 Interface Module The interface module is responsible for carrying out an actual serving event. During a serving event the interface module is given complete control of the robot, and the boss module, explained below, waits until the interface module indicates that it is finished. The modular structure of the robot software allows for different interface modules for each of the two events of the robot host competition: serving desserts and serving information. Both communicate with the other modules via IPC.

A serving event can be initiated either by the robot's boss module, or by detecting a person while the interface module is in the idle state, such as when the robot is in the wander mode. In the former case, the boss module sends a message to the interface module when the robot has located and approached a person. This message indicates that the interface module should offer to serve the person. The boss module then waits until the interface module sends an acknowledgement to indicate that the serving event is over. When the interface module is idle it listens for an event on the keyboard, in the information-serving module, or in the case of dessert-serving listens for a message from the vision module indicating that the vision module has seen a hand reaching for a cookie.

The interface module is a state machine, and once initiated it progresses through various states to complete a serving event. This often involves waiting for the person to do something such as select a menu option by pressing a key on the keyboard, but the module always keeps a timer

which times out if it receives no response in a fixed period of time. This keeps the robot from getting stuck waiting for input when the user has walked away. Both interface modules communicate with people by sending appropriate text to the speech module for speaking, and the information-serving module also made use of the LCD screen.

The weak point of the interface module was the information database, which was implemented as a simple text-based menu-driven system accessing a very limited amount of information.

3.1.5 Boss Module The Boss module coordinates the states and behaviors of each of the other modules. On start-up, Boss reads in a configuration file that lists all of the modules that should be started. It starts the IPC Central Server and then initializes each of the modules. During a run, Boss listens for messages from all of the other modules.

Boss is the only module that sends state change commands to the other modules. It listens to the Vision operator data to determine when a person is present, then instructs the control module to approach the person. When a person to serve has been identified, Boss tells the Interface module to change to the Serve state. Similarly, it is the Boss module that watches for the other robot's Italian flag so that the Speech module can be instructed to start a conversation.

The Boss module starts the events by randomly selecting an area of the room to cover. It then enters the Wander Achieve mode, which allows the Con module to take advantage of obstacle avoidance strategies while aiming primarily for the target area. Once the area is reached, the robot wanders for a fixed period of time. Only after the robot has been wandering for long enough will it begin to explicitly approach people to offer them information. At the same time, the Boss module keeps track of how much area it has covered in the current region; if it has been in a small are for too long it will pick a new region to approach and wander.

Because the serving area is so crowded, and because one of the primary goals of the robot host competition is to cover a large area during the serving time, the Boss module spends a good part of its time in the Wander state, in which the Con module uses basic obstacle avoidance to move through the room without explicitly approaching anyone to offer information or desserts. If someone interacts with the robot while it is in the wander state--either by hitting a key on the keyboard for the information task or by taking a snack in the food task--then the Interface module will notify the Boss module of the event and the robot will stop and interact with the person.

Once the robot has wandered sufficiently, it will begin to look for people to serve. The Boss module requests information from the vision module on the location of faces and pink ribbons--which are used to identify participants in the robot competition. At fixed intervals, the Boss module will also have the robot stop and try to detect motion around it. Upon finding a person, the robot will begin to approach the location that it thinks someone is standing. When its sensors indicate that something is close by, it will again check for motion to make sure it has found a person before offering to serve the person. If the robot moves more than a fixed distance in the direction that it thinks it saw a person without encountering anyone, the command to approach the person will timeout and the robot will revert to the Wander state.

Since the Boss module waits for acknowledgements from the other modules to change out of some states--like serving and conversing--a crucial element of our design ended up being the addition of timeout messages. If a module is unable to complete an action in a given amount of time, Boss will return everything to a default state and move on. This keeps the robots from getting stuck if, for example, an unexpected input was able to freeze the information-serving interface, or if one robot's battery dies while the other robot is waiting for an acknowledgement from it in a conversation.

3.2 USR

For the past two years, Swarthmore's USR entries have combined autonomy with tele-operation to generate semi-autonomous systems. The goal is to use the best features of both forms of control. The robot possesses quicker reactions and a better sense of its immediate environment, while the human operator has a better sense of where to go, what to look at, and how to interpret images from a camera.

Our USR system gives the operator the ability to specify relative goal points, stop or orient the robot, and control a pan-tilt-zoom camera. The robot autonomously manages navigation to the goal point using a reactive obstacle avoidance system. Giving the robot reactive control turned out to be extremely important, because the robot was able to sense things in the environment that were not perceptible to the operator, such as transparent surfaces.

Having a control system that permitted quick reaction times for both the operator and the robots was a primary focus this year. Using IPC for communication both reduced the lag time and increased the frame rate of images coming from the robots' cameras across the wireless network as compared to last year's X forwarding. This year's system also gave the operator more immediate control over the type of image stream coming from the robot, permitting quarter size and half size greyscale or color image streams at the touch of a button.

As in 2001, Swarthmore used two similarly equipped robots, primarily using one robot to watch the other as they traversed the course. This turned out to be helpful in overcoming obstacles that were not readily apparent from the point of view of the lead robot. A secondary purpose of

Figure 3: A video feed from the camera attached to each robot allows the operator of the robots to assist in the victim-detection task.

using two was to have a spare in case of equipment failure. This turned out to be critical in the final run—when Swarthmore had its best score--as the lead robot's camera control cable failed. The trailing robot continued on and proceeded to find three more victims. Such a scenario is not unreasonable to expect in a true USR situation.

3.2.1 Overall architecture The modularity of our system design allowed us to build our USR system from the same components that we used in the robot host competition. For USR, the vision and control modules were relevant. The vision module was used to monitor the video coming from the cameras on the robots for signs of victims. Both the Face and motion operators were useful in locating victims. The findings of the operators were marked with colored squares on top of the video feed from each robot's camera, which allowed the operator to use the combined input of the camera and the vision operators to determine the location of a victim.

3.2.2 Mapping Module In addition to the vision and con modules, the USR event also used a map module that provided graphical maps from the start location to the current location of the robot. The basis for the maps was an evidence grid built from the sonar readings and odometry information [5]. This provided a general outline of the rooms and obstacles encountered by the robot.

In addition, the mapping module builds a set of landmark points in the map such that each landmark point has a clear path to at least one other landmark point. Upon request, the mapping module uses Dijkstra's algorithm to calculate the shortest path between the start location of the robot and the robot's current position. By overlaying this path on the evidence grid, the robot is able to generate maps to victims it finds during the USR event.

3.2.3 Interface Module One of our goals this year was to replace the command-line interface we have used previously with a graphical user interface for controlling the robots during the USR competition. In this way, we hoped to make it easier for a single operator to monitor and instruct both robots simultaneously.

The interface is written in C using the Motif toolkit and is designed to control two robots simultaneously. Each robot's panel is divided into a top half and a bottom half; each half consists of a display with surrounding buttons. The top half deals with the vision module: its display shows the video feed from the robot's camera while the buttons surrounding it control settings pertaining to the vision module and the camera.

The buttons to the left of the video display set the video modes--small or medium sized video feed, color or gray-scale—while the buttons to the right set pan, tilt, and zoom parameters for the camera. Buttons below the display turn various operators on or off.

The lower half of the panel deals with navigation and localization. The display shows the map built by the map module, while buttons to the left and right of it request information from the map module and scroll the displayed map. Buttons below the display control the robot's movements.

We tried using the new interface in one of our test runs and found that it was simply not as responsive as some of the command-line programs that we used to test each of the components of the system. While the interface worked well in testing, the response time when both of the robots as well as the computer running the interface were all relying on wireless network connections was unacceptable. For each of our official runs, we chose to run several different test programs simultaneously to view the video data from each camera, monitor the vision module results, switch between manual and autonomous navigation, and steer the robots. While this resulted in a far less elegant interface--with each piece running in a separate X-window--the speed increases allowed us to successfully navigate the robots simultaneously.

4 Review of performance and future goals

While Frodo and Rose's average performance in the Host competition was somewhat disappointing, we felt that they had the best single run of all participants in the last round of information serving, and a very solid performance dessert-

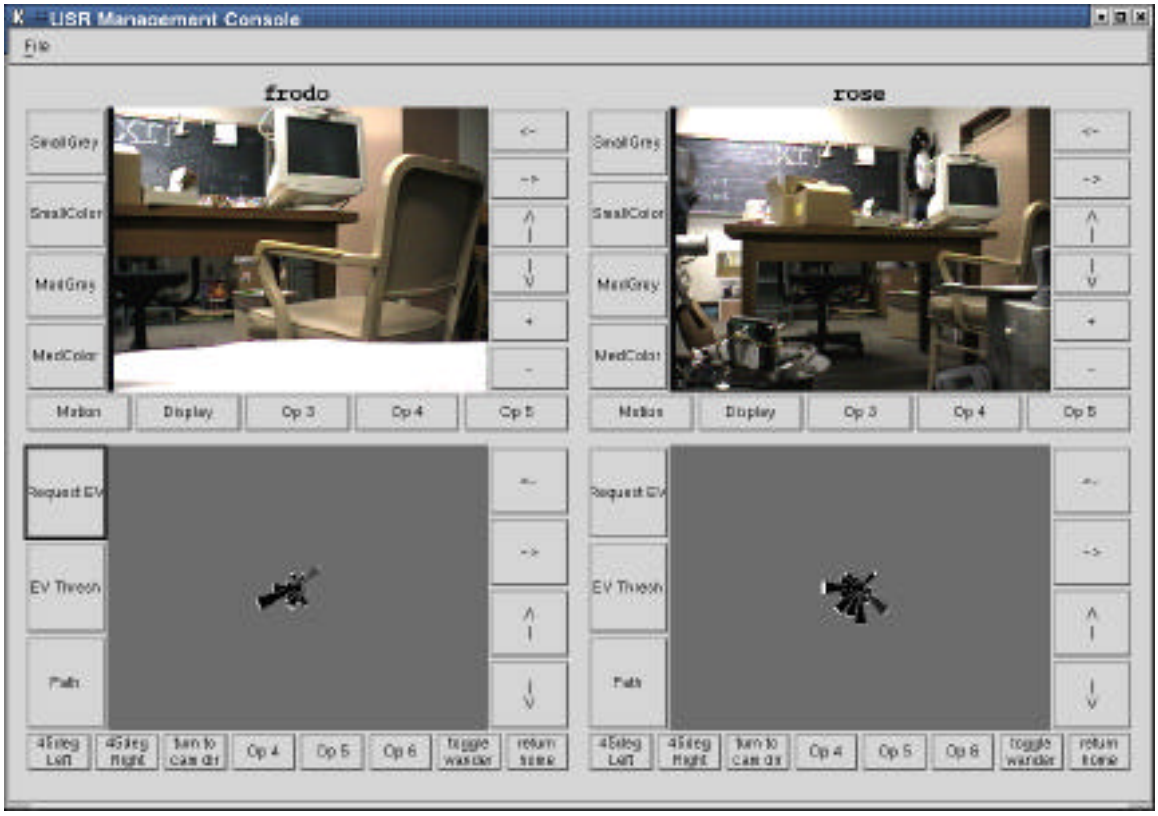

Figure 4: The graphical user interface allows a single operator to easily monitor and control two robots.

serving. The default wander behavior coupled with the mapping functions made for superior coverage of the very crowded competition area. The timeouts proved to be very important to robust behavior. As stated above, the primary weakness during the information serving was the actual information database. In general, more preparation time would have been helpful in developing the robot software. We also found that all the peripherals—especially the LCD display—contributed to an energy drain and extra weight that taxed the robots' batteries to the maximum: we only got about 40 minutes of power during the information serving events before running the batteries completely down. As this was the last year for the Robot Host competition, it is somewhat moot to ponder specific future goals for that task.

A primary goal for next year is to make the graphical interface for our USR system useable in a real-time environment. While we were successful—placing second out of nine—the command-line interface tools used to operate the robots in the USR event are somewhat cumbersome, and certainly not up to the standard goals of layman usability.

References

[[1] B. A. Maxwell, L. A. Meeden, N. Addo, L. Brown, P. Dickson, J. Ng, S. Olshfski, E. Silk, and J. Wales, "Alfred: The Robot Waiter Who Remembers You," in *Proceedings of AAAI Workshop on Robotics*, July, 1999.

[[2] B. A. Maxwell, L. A. Meeden, N. S. Addo, P. Dickson, N. Fairfield, N. Johnson, E. Jones, S. Kim, P. Malla, M. Murphy, B. Rutter, and E. Silk, "REAPER: A Reflexive Architecture for Perceptive Agents," *AI Magazine*

[[3] R. Simmons and D. James, "Inter Process Communication: A Reference Manual," February 2001, for IPC version 3.4. Carnegie Mellon University School of Computer Science/Robotics Institute.

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

[[5] A. Elfes, "Sonar-Based Real-World Mapping and Navigation", *IEEE Journal of Robotics and Automation* 3:3 pp. 249-265, 1987.