Blue Swarm 3: Integrating Capabilities for an Autonomous Rescue Robot Swarm

Daniel P. Stormont, Anisha Arora, Madhumita Chandrasekharan, Manasi Datar, Udit Dave, Chaitanya Gharpure, Juan Jorge, Aliasgar Kutiyanawala, Vinay Patel, Bharath Ramaswamy, Loren Sackett, and Zhen Song

Departments of Computer Science, Electrical and Computer Engineering, and Mechanical and Aerospace Engineering
Utah State University
Logan, Utah 84322
stormont@cc.usu.edu

Abstract

The robotics team at Utah State University has been working on incrementally building the capabilities needed to field an autonomous swarm of rescue robots. We believe a swarm of small, low-cost robots that can identify areas needing further investigation would be of great benefit to first responders to the scene of a disaster. Blue Swarm 3 is the latest iteration in this development process and is the first swarm to try to incorporate all of the capabilities needed for a fieldable rescue robot swarm. This paper discusses the philosophy behind Blue Swarm 3, the historical background behind the previous swarms, details of the design of Blue Swarm 3, the results of our efforts at this year's American Association for Artificial Intelligence Mobile Robot Competition, lessons learned, and plans for the future development of the Blue Swarm 3 and follow-on swarms.

Introduction

Autonomous robots are not commonly used in real-world search and rescue operations today. Given the current state-of-the-art in autonomous robotics, rescuers will continue to rely upon tele-operated robots that make use of the unparalleled processing power of the human brain and upon trained animals, such as search and rescue dogs. However, we believe that there is a place for autonomous robots in search and rescue operations. A swarm of small, low cost robots could provide an invaluable "force multiplier" for first responders to a disaster area by providing them with an overall view of the disaster scene and highlighting areas that should be the most productive to begin searching with the other tools at their disposal – such as tele-operated robots, dogs, and human rescuers.

To provide this capability, the robotics team at Utah State University has been attempting to develop a swarm of autonomous search and rescue robots with the objective of keeping the cost low enough that rescuers wouldn't feel the need to recover them from the disaster area, small enough that they wouldn't cause harm to victims or rescuers, and capable enough that they can provide useful information via robust communications channels to first responders. To say that this is a challenge would be an understatement. This paper describes the current status of our efforts.

Historical Background

The American Association for Artificial Intelligence (AAAI) introduced the rescue robot competition as part of their annual mobile robot competition in 2000. This competition utilizes the standard test course for urban search and rescue developed by the National Institute for Standards and Technology (NIST). The NIST test course provides a variety of simulated disaster situations varying from minor damage and debris in the yellow arena, through more extensive debris and multiple levels in the orange arena, to a rubble pile in the red arena. Utah State University has been involved with the AAAI rescue robot competition since it was first introduced. Each year, we have focused on a different capability that would be needed to eventually field an effective swarm of autonomous rescue robots. The following paragraphs describe these efforts, the capabilities being focused on, and the results.

Blue Swarm

Our initial effort in 2000 was to modify six remote-controlled toy cars with a simple analog control circuit to determine the lower boundary in cost and capability. The control circuit was the tutebot circuit described in (Jones, Seiger, and Flynn 1999). This simple analog circuit was only capable of responding to obstacles in the environment. It was unable to detect a victim, determine its position, or report a victim location to a rescuer. Our objective was limited to determining if this simplest of robots could fully explore the simplest arena in the urban search and rescue testbed: the yellow arena. Unfortunately, the Blue Swarm was hampered by problems with the control circuitry. The motor speed was too difficult to control since it was adjusted via a potentiometer setting prior to deploying the robot in the arena. The relays that controlled the motor directions also failed frequently. As a result, the Blue Swarm never ran in the competition – it was only displayed in the exhibit area, as shown in figure 1.

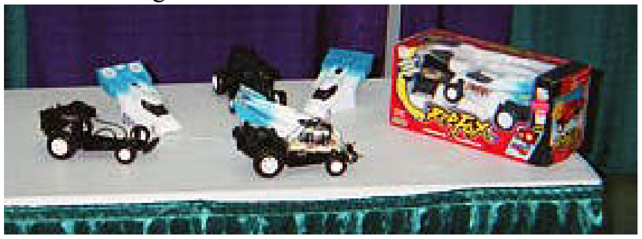


Figure 1. The Blue Swarm robots on display at the AAAI 2000 Mobile Robot Exhibition.

Blue Swarm 2

The shortcomings evident with the Blue Swarm led us to increase the complexity and capability of the robots used in the 2001 competition. The same type of modified toy cars were used as the chassis for the robots, but a microcontroller replaced the analog control circuit. The objective was to see how much area the six robots in the swarm could cover in the time allotted for a competition run. The robots also had an IR-based temperature sensor to try to detect victims, but they still had no capability to determine their position or the position of a located victim. They also were not capable of reporting the position of a victim to the rescuers. The six robots that comprised the Blue Swarm 2 were consistently able to explore all of the accessible areas in the yellow arena. In fact, the area covered by these six robots using a simple random walk navigation algorithm frequently was greater than the area covered by the tele-operated robots that year. The victim sensor circuit did not work, so the robots were unable to signal that they had located a victim. The Blue Swarm 2 robot, shown in figure 2, was described in (Boldt and Stormont 2001).

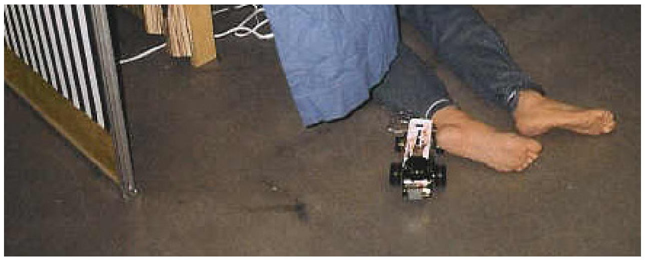


Figure 2. The Blue Swarm 2 prototype appears to have found a victim in the yellow arena. It was actually unable to detect victims because the victim detection sensor did not work.

Blue Swarm Sentinel

In order to address the shortcomings of not being able to determine the location of a potential victim and not being able to report that position to the rescuers, the Blue Swarm Sentinel was designed as an adjunct to the Blue Swarm 2 for the 2002 competition. The Blue Swarm Sentinel was based on a radio-controlled toy tank. The radio control circuitry was replaced by a network of three microcontrollers. The Sentinel had a wireless video camera to display images on the rescuer graphical user interface (GUI), an infrared temperature sensor for victim identification, a compass module and encoder switches on the treads for pose estimation, collision avoidance sensors, and an RF transmitter-receiver pair to communicate with the GUI on the base station. The robot could operate in both manual mode so the rescuer could drive the robot to a desired starting location and autonomous mode to explore the environment. The Sentinel was described in (Bhatt et al. 2002) and is shown in figure 3.

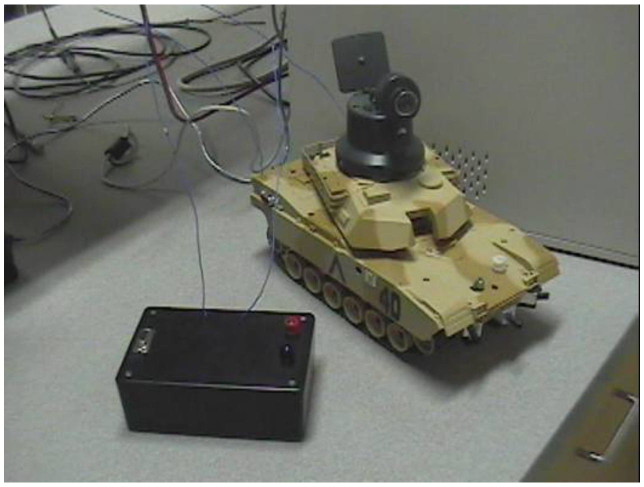


Figure 3. The Blue Swarm Sentinel and its base station unit.

The Sentinel was not used in the competition because of failures in the radio modules and interprocessor communications. Instead, a second RC tank was converted into the “breadboard special” by adding two breadboards and a transceiver module. This robot ran in the competition, but was very limited in sensor capability and frequently got stuck on obstacles like chair legs. A picture of the breadboard special is in figure 4.

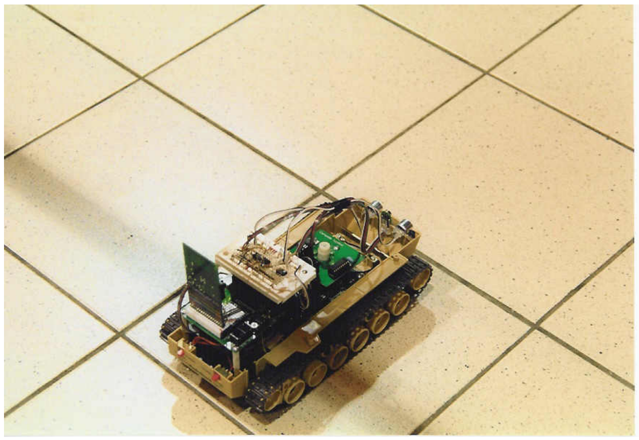


Figure 4. The "breadboard special" prepares to enter the yellow arena.

Blue Swarm 2.5

The swarm for 2003 was intended to test some concepts in sensor fusion for the rescuer GUI, as described in (Stormont and Berkemeier 2004), for possible incorporation into the next iteration of the Blue Swarm. Since it wasn't actually an increment in the development of the Blue Swarm, it was given the designation Blue Swarm 2.5. To collect the sensor data, two Boe-Bots from Parallax were modified with crawler kits for added mobility and equipped with a number of victim sensors; including a CMOS color camera, ultrasonic sensors, infrared rangers, an infrared temperature sensor, and a compass module. The Blue Swarm 2.5 was described in (Stormont and Berkemeier 2003). In the competition, the microcontrollers kept running up against memory constraints, which limited the use of the sensors and made it impossible to transmit data when the robots were moving, so the desired sensor data was not collected. Figure 5 shows a Blue Swarm 2.5 robot in the rescue arena.

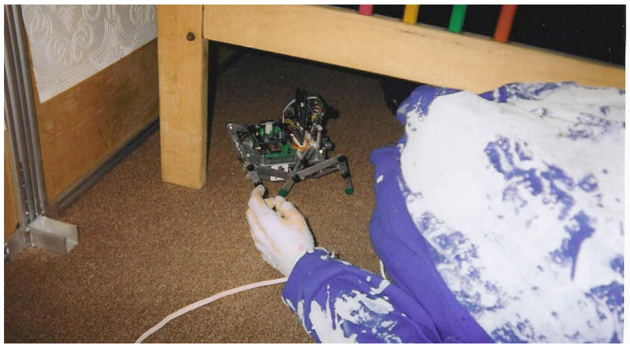


Figure 5. A Blue Swarm 2.5 robot approaches a victim in the yellow arena.

Blue Swarm 3

Conceptual Approach

This year's entry was the first to try to bring together the capabilities tested piecemeal in the previous incarnations of the Blue Swarm. Blue Swarm 3, shown in figure 6, consists of ten robots with a full suite of sensors for navigation, localization, and victim detection. The swarm also has two means of communicating information back to the rescuer GUI: radio frequency and line-of-sight using infrared emitters and detectors. As currently implemented, the robots in the swarm form an ad-hoc sensor network by maintaining a neighbor relationship with no more and no less than two other robots in the swarm. Maintaining this relationship forces the robots to spread out into the rescue arena, while maintaining line-of-sight contact for localization and communications. The hardware and software architecture designed for Blue Swarm 3 is robust and flexible enough that we expect it to be our platform for experimenting with autonomous search and rescue concepts for several years to come. The following paragraphs provide details about the design of the Blue Swarm 3.

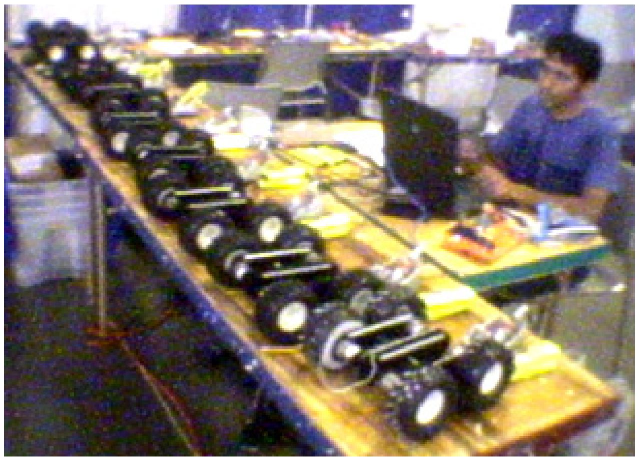


Figure 6. The Blue Swarm 3 robots being worked on in the pit area at AAAI 2004.

Hardware Design

The Blue Swarm 3 robots make use of commercially available parts wherever possible. By necessity, some of the controller and sensor boards were custom-made, but everything else was purchased from various robot vendors. Each subsystem of the robot has its own controller. Figure 7 shows a block diagram of the robot hardware, with dashed lines separating the subsystems: primary controller, localization, local mapping, victim detection, and communications. Figure 8 shows an image of a Blue Swarm 3 robot.

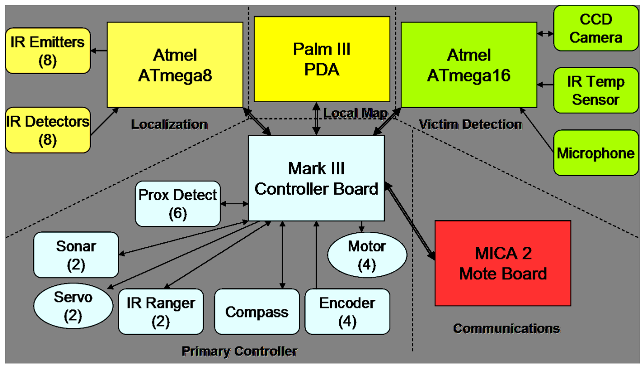


Figure 7. Hardware block diagram of the Blue Swarm 3 robots.

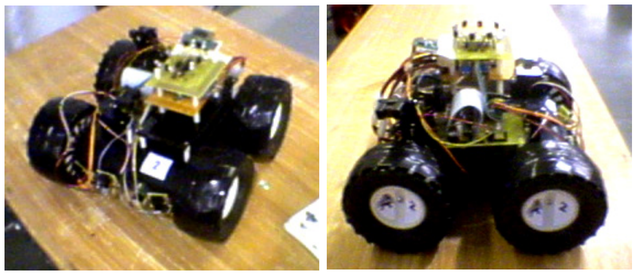


Figure 8. Two views of one of the Blue Swarm 3 robots showing the collision avoidance sensors in front, infrared ranging sensors on the front and back servo motors, and the localization sensors on the top circuit board.

Robot Chassis. The Blue Swarm 3 robots use the 4WD2 articulated four wheel drive robot chassis from Lynxmotion. We chose this chassis because the articulated hulls appeared to provide greater obstacle climbing capability than a chassis without an articulated hull, as shown in figure 9. The locomotion for the robot is provided by four 7.2 VDC motors with 50:1 gearboxes.

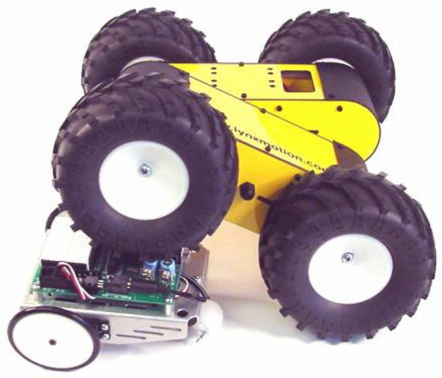


Figure 9. The Lynxmotion 4WD2 chassis demonstrating the articulation of the hulls. (Image courtesy of Lynxmotion.)

Primary Controller and Navigation Sensors. We selected the OOPic Mark III robot controller board as the primary controller board for the robots. The OOPic microcontroller is a good choice for the low-level functions of the robot because of its event-based multi-tasking architecture and because it has a number of objects (called virtual circuits in the OOPic) that support common sensors, actuators, and communications protocols like RS-232 serial and Inter-IC (I2C) protocols. The Mark III controller also has the advantage of being small and inexpensive. Headers allow access to all of the pins on the microcontroller, making interfacing peripherals to it much easier.

The function of the primary controller is to route all of the communications between subsystems on the robot and to move the robot by sending control signals to the motor driver circuits while the primary controller monitors the navigation sensors. The navigation sensors consist of six infrared proximity detectors for collision avoidance (three facing forward in an arc, two facing down in front of the front wheels, and one facing down in the back of the robot), four infrared photodetectors and segmented encoder disks for wheel velocity measurement, and an 8-bit compass module for heading determination. The primary controller also controls the mapping sensors, which consist of an infrared and an ultrasonic rangefinder on a servo motor with a 180° range of motion mounted on either end of the robot.

Localization Sensors. The localization sensors are eight infrared emitter/detector pairs arranged in a circle so that they divide the 360° around the robot into eight 45° sectors. They are controlled by an Atmel ATmega8 microcontroller on a custom-made circuit board. We selected the ATmega8 because of its low cost, ample memory, high speed, and sufficient number of I/O pins for the task, as well as for a wide range of low-cost and open source development tools. The localization sensors perform the task of determining the direction of other robots in the line of sight of the robot and double as a secondary communications path.

Victim Detection Sensors. The Blue Swarm 3 robots use three sensors for detecting victims: a CMOS camera, a microphone, and an infrared temperature sensor. The CMOS camera is the GameBoy camera made by Nintendo. The CMOS camera chip inside the camera handles much of the image processing, such as inverting video and edge detection, thus limiting the processing required. We selected an Atmel ATmega16 microcontroller for the victim detection function. It is fast enough and has enough memory to handle the nearest object determination task for the camera and to listen for non-repetitive sounds from the microphone. It has I2C communications capability, so it can communicate with the temperature sensor, which provides ambient temperature and the temperature of any objects in front of the sensor, up to about a meter away. It also shares common development tools with the ATmega8 used for the localization sensors. The victim detection sensors perform the function of identifying objects that may be victims and then trying to gather enough sensory data about the objects to determine if they are a victim or an obstacle.

Local Mapping Controller. Every robot keeps a map of the path it believes it has followed and any obstacles or victims encountered, based on its sensor information. This map is kept on a Palm III Personal Digital Assistant (PDA). The Palm III was selected because it is inexpensive, has sufficient memory for building a local map, and uses a serial interface so it can communicate with the OOPic microcontroller via the Serial Communications Protocol (SCP). One advantage of keeping a map on the PDA is that the local map provides a back-up method of constructing a global map of the disaster area, if the robot can be recovered.

Communications. The communications subsystem is based on the MICA 2 mote board from Crossbow. The mote boards are based on the Atmel ATmega128 microcontroller and Intel's smart mote communications chips. The mote boards use open source software developed by the University of California at Berkeley called TinyOS to create an ad-hoc TCP/IP network. The ATmega128 microcontrollers have plenty of memory space for buffering processed sensor data, so even relatively long periods of real or simulated RF signal loss can be tolerated. The primary responsibility of the communications subsystem is to provide localization, victim, and pose information to the rescuer GUI, which uses data from all of the robots in communications with the base station to try to build a global map of the rescue arena.

Software Design

The environment in the urban search and rescue testbed is highly unstructured and unknown to the robots in advance. Even a map created in one run in the arena will not be valid for the next run since even major structures, such as walls may have been moved. Because of the unknown and unstructured environment in the competition and in the real world, we think a reactive behavioral architecture is the most robust and the most adaptive for this application. Therefore, we designed our software architecture using the subsumption architecture first proposed by Brooks (1986). Our software design is illustrated in figure 10. The software architecture is composed of the six behaviors described below, with most of the behaviors being executed on a processor dedicated to that behavior. The only exceptions are the two highest-level behaviors, *build map* and *monitor movement*, which both run on the PDA and the not so easily categorized drivers for processing sensor inputs, which are divided between the primary controller (OOPic), and the localization and victim detection controllers. The rescuer GUI is not shown in figure 10 since it is a coordinating task that attempts to disambiguate the data coming from the robot swarm and produce a coherent global map. It will be described after the descriptions of the behaviors.

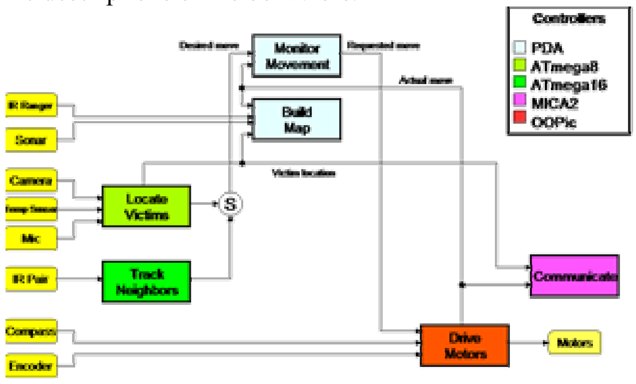
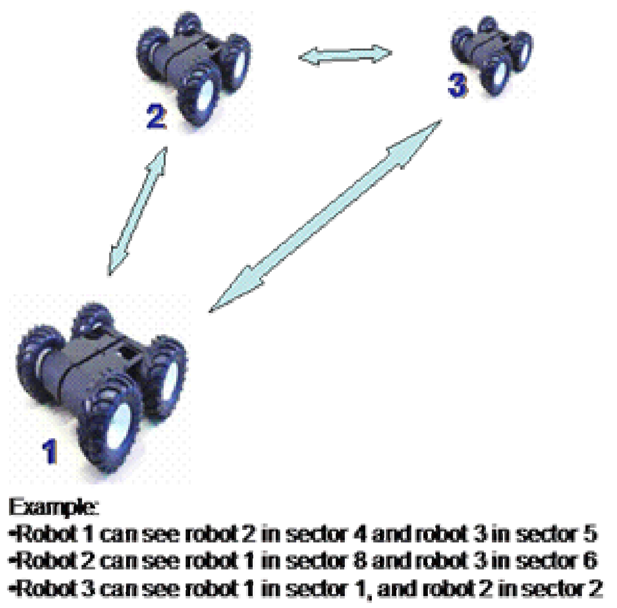


Figure 10. The software architecture for the Blue Swarm 3.

Drive Motors. This behavior receives the requested move from the *monitor movement* behavior in the form (<direction to move>, <number of cells to move>, <final heading>) and attempts to execute the request. It may be unable to execute the request if one of the collision avoidance sensors detects an obstacle that would prevent carrying out the requested move. The *drive motors* behavior uses the compass readings and encoder readings to determine when it has completed the move request. Whether or not the requested movement has been completed, *drive motors* will return the movement that was actually completed, in the same message format as the move request, to the *build map* and *monitor movement* behaviors. The actual move will also be sent to the *communicate* behavior for processing by the rescuer GUI.

Track Neighbors. This behavior has the responsibility for maintaining the neighbor relationships mentioned earlier. Critical to the search algorithm employed for the Blue Swarm 3 is the necessity to always keep no more and no less than two neighboring robots within line of sight. If more than two neighboring robots are within sight, then the swarm has not dispersed as far as it possibly can. If fewer than two robots are in sight, then it will not be possible to use triangulation to attempt to provide relative localization for the three robots and ambiguities in the estimated poses of the robots will be nearly impossible for the rescuer GUI to resolve. This behavior uses time slicing, which requires periodic synchronization pulses from the rescuer GUI to keep the individual robot clocks from drifting too much. Each robot knows its own ID and will broadcast a binary representation of that ID during its broadcast time slice (which is usually on the order of a second). The other robots are all receiving when it is not their turn to broadcast, so they will receive the ID transmission from a broadcasting robot if it is in line of sight. Thus, they have two methods for determining the ID of a neighbor: the ID tag and the time it was received. The signal will be strongest in one sector, so that is the assumed direction of the neighboring robot. The robots can also determine distance to the neighboring robot using the infrared or ultrasonic ranging sensors or a very rough estimate using the localization infrared pairs as a distance sensor. *Track neighbors* maintains a database of the robots it knows a position for and can communicate this information by substituting the database contents for the robot ID, which is an alternative algorithm we have experimented with. *Track neighbors* maintains the desired neighbor relation by sending out a desired move message to the *monitor movement* behavior. This request takes the same form as the move request discussed earlier, except that a final heading is not requested. Figure 11 shows an example of the neighbor relationships *track neighbors* is responsible for maintaining.



Example:
•Robot 1 can see robot 2 in sector 4 and robot 3 in sector 5
•Robot 2 can see robot 1 in sector 8 and robot 3 in sector 6
•Robot 3 can see robot 1 in sector 1, and robot 2 in sector 2

Figure 11. An example of the neighbor relationships maintained by the behavior *track neighbors*.

Locate Victims. This behavior uses sensor data from the CMOS camera, microphone, and infrared temperature sensor to try to locate and identify victims. It primarily uses the camera's edge detection mode to identify objects larger than a threshold value that are close to the robot. The size and distance of the object are determined by the number of horizontal pixels the edge of the object occupies (size) and the vertical position of the edge of the object (distance). This simple determination of size and distance is far from infallible, but it becomes somewhat more accurate since the robot will usually be moving while retaining the same viewpoint. The motion of the robot thus helps to resolve perspective errors. Based on the location of the object in the field of view, *locate victims* will generate a desired move message after every processing cycle in the form (<direction: left, straight, right>, <distance: 1 or 2 cells>). The direction request is based on the location of the center pixel of the closest edge relative to the horizontal center of the field of view and the distance request is based on the number of vertical rows the edge is located above the bottom of the image. If an object is closer than a threshold value, the *locate victims* behavior will take a reading with the infrared temperature sensor and will listen for a non-repetitive noise source with the microphone. If either one or both of these sensors indicate the presence of a victim, the behavior will identify the object as a victim to the *build map* and *communicate* behaviors.

Build Map. This behavior builds a local map based on the movements and sensor readings of an individual robot in the swarm. The map uses an occupancy grid where the individual map cell sizes and the dimensions of the map are based on the expected area of the disaster scene, represented as a two-dimensional array that is twice the expected length and width to ensure the mapping is contained within the bounds of the array regardless of starting position. This somewhat wasteful allocation of memory is acceptable because each individual location in the array only holds a nibble of data indicating an unexplored cell, an obstacle, a victim, or an empty cell. The occupancy grid is also used to generate a graphical local map on the screen of the PDA. The contents of the map array can be downloaded via serial link to the rescuer GUI to generate a global map if one or more robots are recovered from the arena.

Monitor Movement. This behavior acts as the arbitrator of the desired move requests coming from the *track neighbors* and *locate victims* behaviors. It will prioritize the move requests by giving the periodic desired move messages from *track neighbors* the highest priority (to ensure the robots maintain their line-of-sight relationships with their neighbors), followed by the constantly generated desired move messages from *locate victims* if there are no requests from *track neighbors*. However, before converting either one of the desired move messages into a requested move message, *monitor movement* will look at the local map being generated by *build map* to ensure the robot is not stuck in a loop continually exploring the same area. If the desired move messages would result in reentering an already explored area, monitor movement will request a new heading toward an unexplored area and then allow *track neighbors* and *locate victims* to generate new desired move messages, if appropriate. Thus, the *monitor movement* behavior plays a key role in ensuring the robot explores as much of the area as possible.

Communicate. This behavior performs the task of sending information about moves made by the robot, obstacles seen by the sensors, and victims detected to the base station for processing into a global map by the rescuer GUI. It also has the ability to pass along data about other robots contained in the local database generated by the *track neighbors* behavior and to process data or movement requests coming from the rescuer GUI. (This last capability is not implemented at present, but is a growth capability for the future.)

Rescuer GUI. The rescuer GUI is not one of the robot behaviors. It is a stand-alone process that attempts to collect the data about movements, obstacles observed, and victims detected sent by all of the robots in the swarm;

correlate that data; detect conflicting data and ambiguities; and build as accurate a global map as possible. Of course, a high degree of accuracy is impossible with the resolution of the sensors used on the robots and given the unstructured nature of the environment, so the objective of this process is to generate a rough map that highlights areas of interest that can be further explored through the employment of the other tools available to the rescuers, such as tele-operated robots or search and rescue dogs. One of the tools for disambiguating the sensor reports is the localization information provided by the robots. The triangulation between neighbors can be used to identify pose and sensor errors being reported by one of the robots. The resulting global map is displayed to the rescuers using color coded grid cells for explored areas, obstacles, and victims. Certainty measurements could also be taken into account in the color coding of the display, as described in (Stormont and Berkemeier 2004). Additionally, the estimated current positions and headings of the robots are displayed to the rescuers. The GUI has provisions for requesting a still photograph from one of the robots to allow for human interpretation of a victim identified by the *locate victims* behavior. This capability is still being developed.

Competition Results

Most of the behaviors described above were developed and tested to at least a rudimentary extent by the developers working on the individual subsystems. A number of factors complicated the integration of the subsystems, including receiving funding with short deadlines for committing the funds, leading to insufficient prototyping and frequent design changes when problems were identified; delays in organizing the team due to academic commitments during the school year; and finalizing the design late in the design cycle. This last factor was probably the most critical, since it meant some of the most important hardware, such as the custom printed circuit boards, was completed just before the team departed for the AAAI mobile robot competition. Not surprisingly, problems with the circuit boards were uncovered at the competition, necessitating some last minute construction of replacement boards. There were also some unexpected problems, such as the inability during subsystem integration to get the OOPic microcontroller to communicate with the Atmel microcontrollers via the I2C protocol. Since this was a critical element of the hardware design and had not been a problem with most of the sensors, which also communicate via I2C, trying to find alternative inter-processor communications methods was essential, but ultimately unsuccessful. Finally, we had problems with version control, where changes made to previously working software would result in less functionality than before the changes were made and the previous version could not be recovered. This was an especially common problem with the OOPic development environment, which acts like an interpreter, allowing downloads of modified software to the microcontroller without saving the changes.

The end result was that we had three opportunities for scored runs in the preliminary rounds of the rescue robot competition. For the first run, none of the robots was able to run. For the second run, we had one robot ready to run, but the collision avoidance sensors were not responding properly and the robot would collide with debris at the end of the entrance hallway and get stuck, as shown in figure 12. For the third run, the problem with the sensors was fixed but loose connections on one side of a motor driver board we had built on site caused the motors on the right side of the robot to shut down intermittently and the robot to arc into the wall of the arena. In short, we were never able to demonstrate the capabilities we had been developing for the Blue Swarm 3 due to a number of hardware and design issues encountered during the competition.

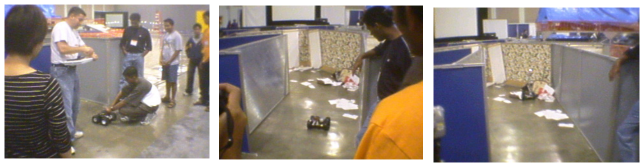


Figure 12. A sequence of photographs showing a Blue Swarm 3 robot colliding with debris in the entryway to the yellow arena during the second preliminary round.

Lessons Learned

The lessons learned from this first attempt at fielding the Blue Swarm 3 echo the lessons learned from many a troubled or failed project. We learned that you can never spend too much time in design and prototyping. We learned that we should have been doing integration testing all throughout the development cycle. We learned that we needed more coordination between team members during the development of the subsystems and the software. Finally, we reinforced the common wisdom that the first version of a circuit board will always contain errors and that failure to implement good version control will always cause problems in software development.

Future Work

Although the competition results for the Blue Swarm 3 in this year's mobile robot competition were disappointing, we remain convinced that we have a good platform for swarm development in the hardware and software designs described in this paper. For the short term, we intend to return to the prototyping phase to develop two or three prototypes for a demonstration at Utah State University in October. The prototype that performs best in the demonstration runs will form the basis for the redesign of the swarm robots. We will then replicate nine more copies of the successful prototype and complete the software development and integration. We hope to enter the rebuilt

Blue Swarm 3 in the 2005 RoboCup American Open Rescue Robot competition and in the AAAI 2005 Rescue Robot competition.

Over a longer term, we would like to incorporate some of the software enhancements that were not a part of this development. Some examples are a robust sensor fusion algorithm that provides a degree of confidence measure in the graphical display on the global map. This could be displayed in the form of varying levels of shading for the colors representing obstacles, victims, and open areas corresponding to the degree of confidence. Another enhancement to the rescuer GUI would be the incorporation of views that would be appropriate to the needs of the rescuer. In other words, a rescue team preparing to enter the disaster area would probably find a three-dimensional walkthrough of what they could expect to find along the path they plan to take more useful than a two-dimensional global map. The ability to cue regions of the global map to try to get the robots close to that area to explore the cued area more thoroughly would also be useful. Desirable hardware enhancements include more accurate position sensors, stereo vision, and higher resolution rangefinders. The Blue Swarm 3 should also provide a useful platform for experimenting with other search strategies, such as formation sweeps, varying numbers of robots comprising a neighbor relationship, and the return of robots to the starting point for recovery of their local maps.

Eventually, it is hoped that the experience gained from the development of the many iterations of the Blue Swarm can be put to use in developing a truly fieldable swarm of autonomous search and rescue robots.

Acknowledgments

We would like to thank the Dean of the College of Engineering at Utah State University for providing funding for Blue Swarm 3. We would also like to thank the Department of Electrical and Computer Engineering for providing additional funding for robot parts and travel to San Jose. The Center for Self Organizing and Intelligent Systems (CSOIS) at Utah State provided workspace and tools. We are especially grateful to PCBexpress for their education sponsorship program that enabled us to order our printed circuit boards for only the cost of shipping. Finally, the primary author (Dan Stormont) wishes to thank the Space Dynamics Laboratory at Utah State University, whose Tomorrow PhD Fellowship has made it possible for him to be involved with this research.

References

Bhatt, A., Boldt, B., Skousen, S., and Stormont, D. 2002. Blue Swarm Sentinel and Blue Swarm 2. In *AAAI Mobile Robot Competition & Exhibition*, Technical Report WS-02-18, 50-52. Menlo Park, Calif.: AAAI Press.

Boldt, B. and Stormont, D. 2001. Blue Swarm II. In *AAAI Mobile Robot Competition*, Technical Report WS-01-01, 8-9. Menlo Park, Calif.: AAAI Press.

Brooks, R. 1986. A Robust Layered Control System for a Mobile Robot. *IEEE Journal of Robotics and Automation* RA-2 April: 14-23.

Jones, J., Seiger, B., and Flynn, A. 1999. *Mobile Robots: Inspiration to Implementation.* Natick, Mass.: A K Peters.

Stormont, D. and Berkemeier, M. 2003. Blue Swarm 2.5: A Step Toward an Autonomous Swarm of Search and Rescue Robots. In *AAAI Mobile Robot Competition 2003*, Technical Report WS-03-01, 36-40. Menlo Park, Calif.: AAAI Press.

Stormont, D. and Berkemeier, M. 2004. A Robotic Rescue Graphical User Interface Integrating Multi-Robot Sensor Fusion. In Engineering, Construction, and Operations in Challenging Environments: Earth and Space 2004, 153-160. Reston, VA: American Society of Civil Engineers.