Design and Implementation of a Mechanically Heterogeneous Robot Group

Gaurav S. Sukhatme, James F. Montgomery, and Maja J. Matarić

Robotics Research Laboratories
Department of Computer Science
Institute of Robotics and Intelligent Systems
University of Southern California
941 W. 37th Place
Los Angeles, CA 90089-0781

ABSTRACT

This paper describes the design and construction of a cooperative, heterogeneous robot group comprised of one semi-autonomous aerial robot and two autonomous ground robots (Pioneer ATs). The robots are designed to perform automated surveillance and reconnaissance of an urban outdoor area using onboard sensing. The ground vehicles have GPS, sonar for obstacle detection and avoidance, and a simple color-based vision system. Navigation is performed using an optimal mixture of odometry and GPS. The helicopter is equipped with a GPS/INS system, a camera, and a framegrabber. Each robot has an embedded 486 PC/104 processor running the QNX real-time operating system. Individual robot controllers are behavior-based and decentralized. We describe a control strategy and architecture that coordinates the robots with minimal top-down planning. The overall system is controlled at a high level by a single human operator using a specially designed control unit. The operator is able to task the group with a mission using a minimal amount of training. The group can re-task itself based on sensor inputs and can also be re-tasked by the operator. We describe a particular reconnaissance mission that the robots have been tested with, and lessons learned during the design and implementation. Our initial results with these experiments are encouraging given the challenging mechanics of the aerial robot. We conclude the paper with a discussion of ongoing and future work.

Keywords: aerial robot, heterogeneous robot group, multiple robots

1. INTRODUCTION

The last decade has seen an explosion of research[1–6] in robot groups . However, *heterogeneous* robot groups have not been studied in much detail. We consider a robot group to be heterogeneous if at least one member of the group is different from the others in at least one of the following attributes: 1. mechanics, 2. sensing, 3. computing hardware or 4. nature of onboard computation. This relatively loose condition is easily met by the system we describe in this paper. Our experimental testbed (shown in Figure 1) is composed of a robot helicopter and two mobile ground robots, thus making it a morphologically heterogeneous group.

The advantages of using a group of robots to perform coordinated activity have been discussed extensively in the literature. Heterogeneous groups, in particular, allow for the possibility of redundant solutions to a problem as well as a potentially greater degree of fault-tolerance compared to homogeneous groups. However, heterogeneous groups typically involve higher overhead in terms of system maintenance and design. At the University of Southern California Robotics Research Laboratory we have designed a heterogeneous group comprised of a robot helicopter and several mobile robots for reconnaissance and surveillance applications.

In the work described here, we focus on an experimental task loosely motivated by a small unit operations (SUO) application where a single person controls the overall operation of the system and tasks the robot group at a high level. The corresponding civilian application is concerned with security in urban areas where a single guard would need to control and monitor several (possibly heterogeneous) robots. In the application described here the robot group patrols an essentially open area (a few obstacles are present on the ground) defined by a perimeter. The

Send correspondence to GSS. E-mail: gaurav,monty,maja@robotics.usc.edu. Portions of this papers have appeared previously in the proceedings of the Association for Unmanned Vehicle Systems International AUVSI '99.

Figure 1. The experimental testbed consisting of the AVATAR (Autonomous Vehicle Aerial Tracking And Reconnaissance) robot helicopter and two Pioneer AT mobile robots

experimental area is outdoors. The perimeter is defined by a set of vertices (in the form of GPS coordinates) which, when connected by line segments, form the boundary of a closed convex irregular polygon. In the work reported here the robot helicopter is flown by a human pilot for purposes of safety and speed. We have previously demonstrated control algorithms for autonomous stable hover of the helicopter, however we rely on a human pilot to transition from a stable hover at one point in space to another. This is discussed in greater detail in Section 4. Both ground robots are fully autonomous in the experiments reported here.

The robots in the experiments cover the area bounded by the perimeter and send images back to a human operator via a wireless video downlink. The operator is able to set high level goals for the ground robots such as "follow the helicopter." When a particular ground robot is given this high level goal, it stops patrolling and follows the helicopter. The motivation behind this is to allow the ground robots to "take a closer look" at areas which the operator finds interesting based on aerial imagery. Another example of a high level goal is "go to target." The operator can (using the interface discussed in Section 3) designate points of interest within the perimeter which serve as area markers for robots to periodically explore. The basic idea behind the implementation of the control programs is to achieve robust functionality without explicit top-down planning. Rather, the overall behavior of the group is the result of interacting control systems that run on the individual robots and essentially allow each robot a high degree of autonomy.

The rest of this paper is organized as follows. Section 2 describes the hardware and software used in this work. It may be noted that the mobile robots are off-the-shelf Pioneers from Real World Interface and involve comparatively less hardware customization compared to the helicopter, which is built from a radio-controlled (RC) kit. Thus we discuss the customization for the helicopter in greater detail in Section 2. Section 3 describes the user interface developed during the course of this work. Section 4 discusses our algorithms for helicopter control and ground robot control. The results of the experiments are presented in Section 5. We close by summarizing the contributions of this paper and enumerating directions of future work. Section 6 is a brief review of literature related to this research.

2. HARDWARE AND SOFTWARE DESCRIPTION

Our research in robotic helicopters began in 1991 with the AFV[7] (Autonomous Flying Vehicle). We transitioned to our second robot, the AVATAR (Autonomous Vehicle Aerial Tracking And Retrieval), in 1994 then to our current robot, the second generation AVATAR (Autonomous Vehicle Aerial Tracking And Reconnaissance), in 1997. The "R" in AVATAR has changed to reflect a change in robot capabilities.

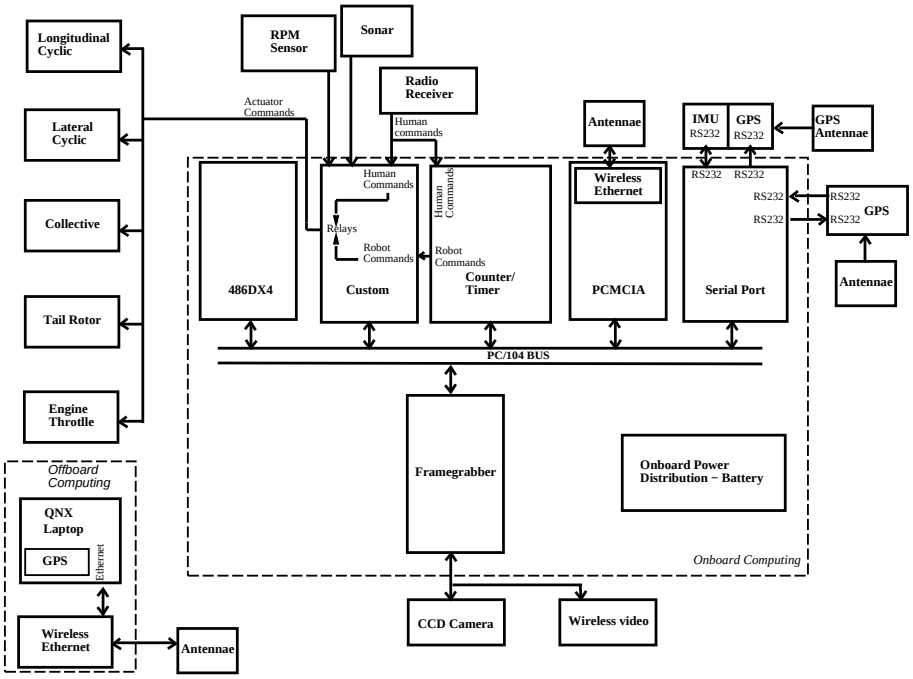

Figure 2. AVATAR System Block Diagram

2.1. AVATAR hardware

The current AVATAR, shown in Figure 1, is based upon the Bergen Industrial Helicopter, a radio controlled (RC) model helicopter. It has a two-meter diameter main rotor, powered by a 4.0 horsepower twin cylinder gas engine, and has a payload capability of approximately 10 kilograms. The helicopter has five degrees of control: main rotor lateral and longitudinal cyclic pitch, tail rotor pitch, main rotor collective pitch, and engine throttle. The first three control the roll, pitch, and yaw of the helicopter while the last two control its thrust. The helicopter can be controlled by a human pilot using a hand-held transmitter which relays pilot control inputs to an onboard radio receiver operating in the 72 MHz frequency band. The receiver is connected to five actuators, one for each degree of control on the helicopter. For autonomous operation, these pilot control inputs are replaced by computer generated control inputs. A block diagram of the AVATAR system; including sensors, onboard and offboard computing resources, wireless communication links, and electrical power sources, is given in Figure 2.

A variety of sensors are mounted on the AVATAR to provide information about the state of the helicopter as well as the environment in which it operates. An integrated Global Positioning System/Inertial Navigation System (GPS/INS) device, consisting of a GPS receiver and an Inertial Measurement Unit (IMU), is the primary sensor used for low-level control of the helicopter. The GPS/INS provides position (latitude, longitude, and altitude), velocity (horizontal and vertical), attitude (roll and pitch), heading (yaw), angular velocity and acceleration information. This particular GPS receiver can only track 4 satellites at once and consequently provides a relatively poor estimate of current latitude and longitude as compared to other available receivers. Thus a second stand-alone GPS receiver is used that can track up to 12 satellites at once. This improves the standard deviations of the estimates of latitude and longitude from 4.5 meters for the 4-channel GPS unit down to 20 centimeters for the 12-channel unit. This GPS receiver is used for the high-level (guidance and navigation) control of the helicopter. A downward facing ultrasonic (sonar) transducer provides altitude information and an RPM sensor mounted on the main rotor mast measures engine speed. A downward looking color CCD camera provides visual information of the area below the AVATAR.

Onboard computing needs are met using a number of commercial off-the-shelf (COTS) PC/104 boards and one additional custom built PC/104 board. PC/104 boards stack together via a header that doubles as a communication and power bus and are useful for embedded applications. The main processor board contains a 486DX4 CPU which runs at 133 MHz, has 16 Mbytes of RAM and 40 Mbytes of solid state disk on chip (DOC). The DOC contains both

the real-time operating system (RTOS) and flight software. The 486DX4 boots up the RTOS at system powerup, executes all flight control and image processing software, and provides an interface to other PC/104 boards.

The PC/104 boards include a timer/counter used both to generate actuator commands and read pilot commands, a custom built board that allows switching between human generated and robot generated commands on an actuator-by-actuator basis as well as acting as an interface to the RPM sensor and sonar, a serial port board for interfacing to the GPS/INS and stand alone GPS, a color video frame grabber for the CCD camera, and a PC/104 to PCMCIA interface board to allow the use of PCMCIA cards onboard the AVATAR. A 2.4 GHz wireless Ethernet PCMCIA card provides a multiway 2.0 Mbps communication link between the AVATAR, other robots, and a human using an operator control unit (OCU). The human operator receives grabbed video frame information and other telemetry from the AVATAR and sends high-level tasking commands to the AVATAR via this link. In addition, differential GPS corrections are sent from the OCU to the GPS receivers onboard the AVATAR through the wireless Ethernet to improve the GPS performance. A live video feed, provided by a one way 1.3 GHz wireless video link from the CCD camera, is displayed on a monitor for viewing by a human.

Nickel-metal hydride (NiMH) and lithium-ion (Li-Ion) batteries supply power to the electronics. Two 10.8 volt, 4.05 amp-hour Li-Ion batteries are connected to a 50 watt PC/104 power supply board that provides +5 volts and $\pm$12 volts to the onboard electronics. The GPS/INS is relatively power intensive, requiring 0.8 amps at +24 volts, thus a dedicated 12 volt, 3.5 amp-hour NiMH battery is connected to a DC/DC converter to produce +24 volt power. The electronics can operate for roughly an hour with this battery supply. Mission length is limited by the flight time of the helicopter on a single tank of gasoline, which is approximately 30 minutes.

2.2. AVATAR software

The operating system used is QNX; a UNIX-like, real-time, multitasking, extensible POSIX RTOS with a small, robust microkernel ideal for embedded systems. The microkernel handles process creation, memory management, and timer control. The flight software encompasses all remaining software running onboard the AVATAR. This currently includes:

- low-level software drivers for interfacing with sensors and actuators,
- flight control software for guidance, navigation, and control,
- learning software,
- self test software including built in test (BIT) and health checks, software for increasing robot fault tolerance
- vision processing software running on the frame grabber,

All software is written in C/C++, with assembly used only when required for speed of execution. Custom drivers have been written for the timer/counter card, the GPS/INS and GPS units, the frame grabber and the wireless Ethernet PCMCIA card. The remaining software is still under development.

2.3. Pioneer hardware

The two Pioneer AT robots (from Real World Interface) used in this work are identical. Each is a four-wheeled base with skid steering. The wheels on the left are coupled mechanically as are the wheels on the right, resulting in two degrees of freedom in the drive. Turning is accomplished by a speed differential between the left and right sides. Each robot has a Lithium-Ion (Li-Ion) battery pack (Two 10.8 volt, 4.05 amp-hour Li-Ion) for the electronics. The motors are powered by a lead acid battery that allows up to 4 hours of operation on hard surfaces and approximately one hour on grass. Each robot is equipped with a ring of seven forward looking sonars controlled by a low-level Motorola 6811 microcontroller. The wheel speeds are available through encoders. The low-level 6811 board is connected to a PC/104 stack via a serial connection. The Pioneer also has a vision system comprised of a camera on a pan-tilt head controlled by a Motorola 68332 board running the Cognachrome color vision system.

The main PC/104 processor board contains a 486DX4 CPU which runs at 133 MHz, has 4 Mbytes of RAM and 40 Mbytes of solid state disk on chip (DOC). The DOC contains both the real-time operating system (RTOS) and control software. The 486DX4 boots up the RTOS at system powerup and executes all control software. Each robot is equipped with a NovaTel GPS system connected to the PC/104 processor via a serial port. Additional sensors

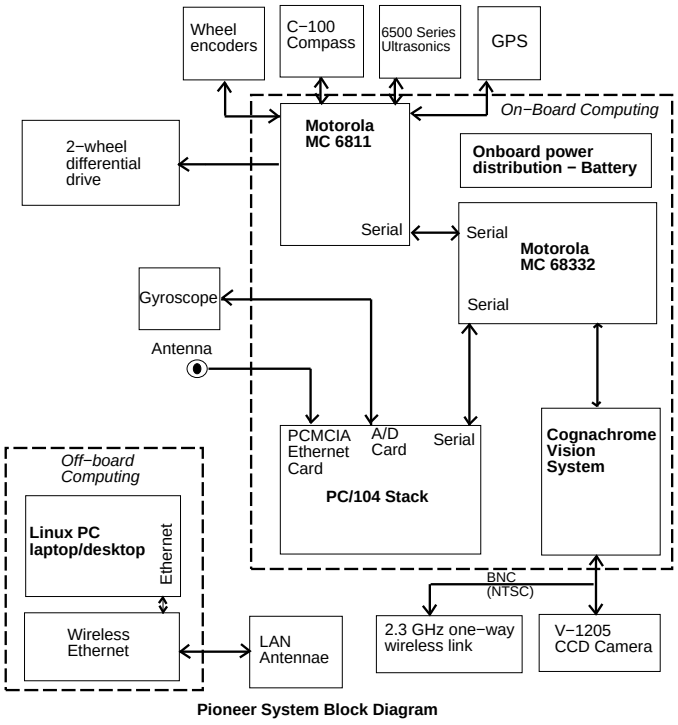

Figure 3. Pioneer System Block Diagram

include a compass and a gyroscope. The gyroscope is connected to a A/D card on the PC/104 stack. A 2.4 GHz wireless Ethernet PCMCIA card provides a multiway 2.0 Mbps communication link between each Pioneer and the other robots. A live video feed, provided by a one way 2.3 GHz wireless video link from the CCD camera, is displayed on a monitor for viewing by a human operator.

A block diagram of the Pioneer system, including sensors, onboard and offboard computing resources, wireless communication links and electrical power sources, is given in Figure 3.

2.4. Pioneer software

The control system for the Pioneer AT (described in the next section) is written in C and runs under QNX on the PC/104 stack described above. The software includes:

- low-level software drivers for interfacing with sensors; specifically software for the wireless ethernet driver and the GPS driver,
- control software for obstacle detection and avoidance, navigation, and mapping.

2.5. OCU hardware

The operator control unit is implemented using a Toshiba Tecra 510CDT laptop PC based upon a 133 MHz Pentium CPU. It has 144 Mbytes of RAM, a 4.3 Gbyte hard drive, a 12.1 inch high-res color monitor, a CD-ROM and an Ethernet connection. The QNX operating system is installed as well as the Watcom C/C++ compiler. A docking station expands the capabilities of the laptop by providing two full-length expansion slots for standard ISA and 32-bit PC Expansion Cards and one half-length 32-bit PC expansion slot. It also has a selectable bay for installing additional hardware such as CD-ROM drives or floppy drives. A 1.44M byte floppy drive has been installed in the slot. The 510CDT has multiple functions and is used during all phases of the project; including development, integration, and testing.

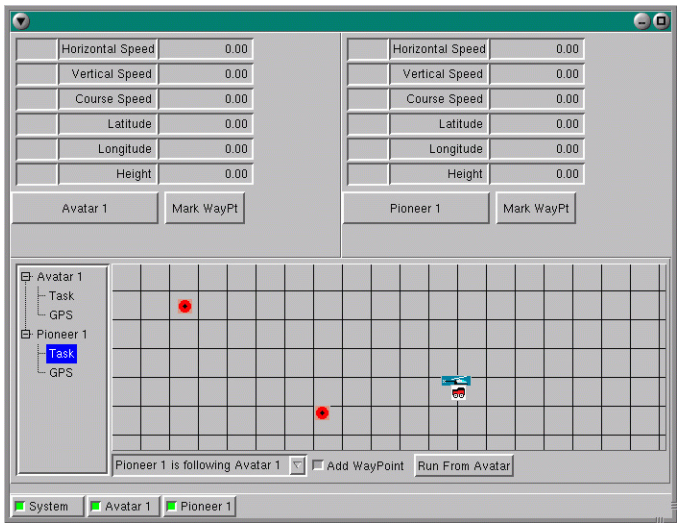

Figure 4. A Screenshot of the User Interface

The primary purpose of the laptop is to function as a wireless OCU for communication and tasking of the robots. A 2.4 GHz wireless Ethernet device is connected to the the Toshiba, providing a multiway connection between the human at the OCU and the wireless PCMCIA cards onboard the robots. The 510CDT provides a software development environment through the use of the QNX operating system and the Watcom C/C++ Compiler for code development and testing, and a software configuration management using RCS, a QNX software version control utility. The 4.3 Gbyte hard drive on the 510CDT provides long-term storage capability for mission data. The docking station provides an ISA slot for a GPS receiver used to produce differential GPS corrections for use by the mobile robots.

3. USER INTERFACE

The interface for the system is implemented under QNX using the Phab GUI development system. The basic layout of the interface is deliberately kept simple so as to allow an inexperienced operator to quickly learn how to use it. A screenshot of the interface is shown in Figure 4.

The user interface allows the operator to examine telemetry from any robot, task individual robots to do specific activities (such as following, patrolling etc.) and monitor the location of the robots in a 2D plan view. In addition, TV monitors show the operator a live wireless video feed from each of the individual robots. Tasking is done by selecting a robot with the mouse and clicking on one of the tasks available in the form of the task list popup menu.

4. CONTROL ALGORITHMS

4.1. AVATAR Control

The AVATAR control system is implemented using a hierarchical behavior-based[8] control system architecture. Briefly, a behavior-based control approach partitions the control problem into a set of loosely coupled computing modules (behaviors). Each behavior is responsible for a specific task and acts in parallel the the others to achieve the overall robot goals. Low-level behaviors in the hierarchy are responsible for robot functions requiring quick response while slower acting higher-level behaviors meet less time-critical needs. The behavior-based control system architecture used for the AVATAR is shown in Figure 5.

At the lowest control level, survival is the main priority. To this end the robot has a set of fast-acting reflex behaviors that attempt to maintain system stability by holding the craft in a hover condition. In a hover, the helicopter maintains a fixed heading and position above the ground. When the robot detects deviations, the appropriate reflex returns the craft to its stable configuration. The *Heading Control* behavior attempts to hold a desired heading by using the IMU heading data to drive the tail rotor actuator. The *Altitude Control* behavior uses the sonar

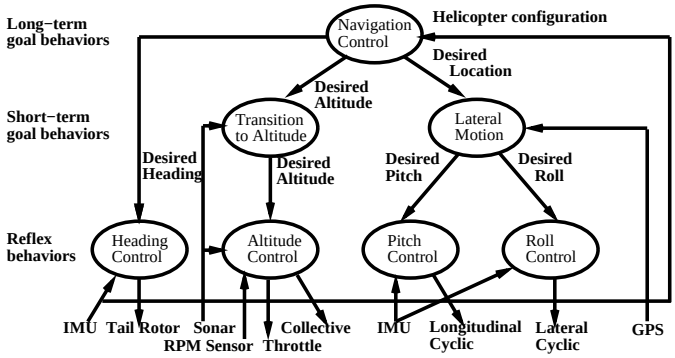

Figure 5. The AVATAR Behavior-Based Control System Architecture

(ultrasonic) data and RPM sensor data to control the collective and throttle actuators. This behavior is responsible for maintaining a desired altitude above the ground. The *Pitch Control* and *Roll Control* behaviors try to hold a stable hover (zero pitch and roll orientation and rate) using the IMU angular orientation and rate data to control the longitudinal and lateral cyclic actuators.

At the next level in the hierarchy, behaviors try to achieve short-term goals of the AVATAR. The *Transition to Altitude* behavior inputs a **Desired Altitude** to Altitude Control to move the robot to a new altitude. *Lateral Motion* generates **Desired Pitch** and **Desired Roll** commands that are given to Pitch Control and Roll Control, respectively, to move to a new lateral position.

At the top level, *Navigation Control* inputs a **Desired Heading** to Heading Control, a **Desired Altitude** to Transition to Altitude and a **Desired Location** to Lateral Motion to navigate the AVATAR to a new heading, altitude, latitude and longitude. Navigation Control produces these desired values based upon tasking commands received from a human using the OCU and the current **Helicopter Configuration** based upon sensory data.

A key advantage of the behavior-based approach is the ability to create greater capabilities for the robot by layering more complex behaviors on top of previously constructed behaviors. This addition is transparent to the lower level behaviors, modulating but not destroying their underlying functionality. This allows the building and testing of control systems incrementally.

We have demonstrated completely autonomous flight on numerous occasions using this control approach with our two earlier robot helicopters, the AFV and the first generation AVATAR. A typical flight would last for five to ten minutes during which the helicopter would maintain desired headings, altitudes, and attitudes. We are just beginning to test with the second generation AVATAR and have demonstrated only autonomous roll control to date. The remaining actuators are currently controlled by a human pilot. As was mentioned earlier, the hardware on the AVATAR allows mixed human/robot operation. This is accomplished through the use of the custom PC/104 hardware that switches between human and robot actuator commands on an actuator-by-actuator basis. We are currently learning the roll control behavior, which will give the robot semi-autonomous control, enabling it to control its roll angle but still requiring a human pilot to control the remaining four actuators.

The reflex behaviors of the two earlier robots were implemented as simple proportional derivative (PD) controllers with gains laboriously tuned through a trial-and-error approach. This was both time intensive and dangerous. The reflex behaviors for the current AVATAR will be implemented as hybrid fuzzy-neural controllers that are learned using an automated "teaching by showing" approach.[9] In this approach, a controller is generated and tuned using training data gathered while a teacher operates the helicopter. No mathematical model of the dynamics of the helicopter is needed. Once the learned controllers meet desired performance criteria, control is turned over to them from the human. We have successfully applied this methodology in simulation, learning pitch and roll controllers capable of maintaining desired pitch and roll angles. We are in the process of validating the approach on the AVATAR. The goal is to eventually learn all of the reflex behaviors, giving the AVATAR complete autonomy.

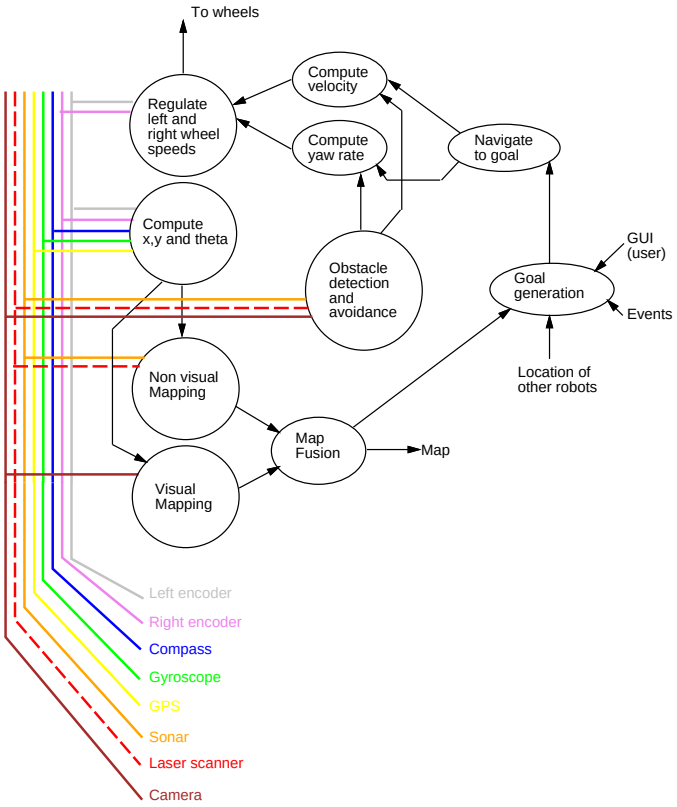

Figure 6. The Pioneer Behavior-Based Control System Architecture

4.2. Pioneer control

The Pioneer robots are controlled using a behavior-based control system depicted in Figure 6. The Pioneer AT robots have four wheels but only two independent controllable degrees of freedom. The wheels on either side of the robot are coupled and are always driven at the same speed. The lowest level behavior is thus to control the speeds of the left and right wheel pairs. This is done by the commercial software available from the robot manufacturer. The commands to this program are generated by our control system.

As shown in Figure 6, the Pioneer robots have a control system that is modeled as a set of interacting processes or behaviors. Sensor input is available to six behaviors. These are: regulate wheel speeds, compute heading and location, non-visual mapping, visual mapping, obstacle detection/avoidance, and goal generation. The only actuator commands generated are the wheel speeds. The laser scanner depicted in the figure is currently not used. The basic control system for each robot also accepts inputs from the base station in the form of user commands (high level tasks) or events (alarms that are triggered by external entities).

We briefly describe below the basic algorithms for each behavior shown in Figure 6. Details about the mapping behaviors are the subject of a forthcoming paper[10] and will not be discussed further here. The "regulate left and right wheel speeds" behavior is provided by the robot manufacturer. The computation of location and orientation[11] is discussed in detail in a separate paper. This computation is basically a Kalman filter operating on five sensor readings: the left and right encoders, the compass, the gyroscope, and the GPS. This behavior enables a ground robot to maintain an accurate estimate of its location and orientation using an optimal combination of these sensors. The particular form of the filter used allows different sensor readings to be collected at different frequencies. Since the filter has inputs from the odometric sensors as well as GPS, the robot is able to keep track of its location and orientation even when GPS is temporarily unavailable (as often happens in urban areas near tall buildings). The obstacle detection and avoidance behavior currently uses the front sonar ring on the robot to roughly locate the

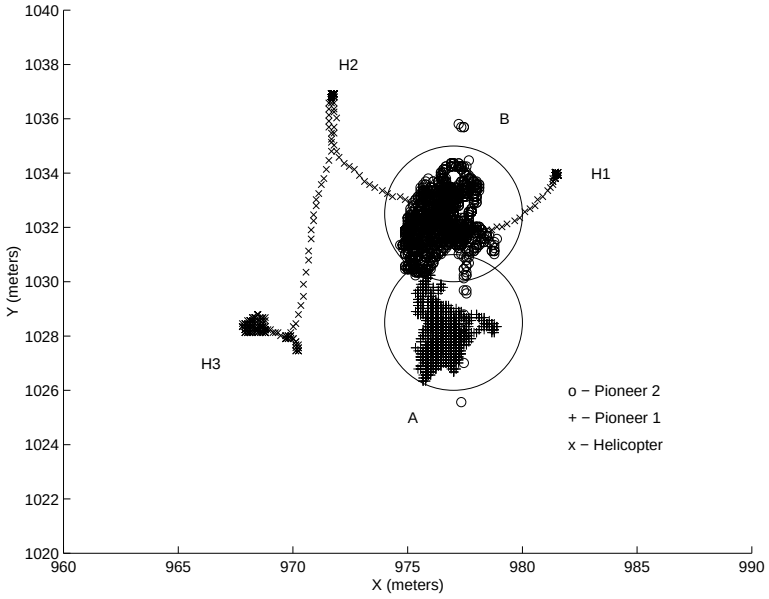

Figure 7. The locations of the three robots over time. The two ground robots are exploring their designated areas of interest (A and B) and the helicopter is hovering at various locations (H1, H2 and H3). The ground robots have not been tasked with helicopter following or alarm response.

angular and radial position of obstacles in a robot-centered reference frame. This information is made available to the behaviors responsible for computing the yaw rate and the velocity of the robot.

The "navigate to goal" behavior is basically a servo-controller on the error in desired orientation and location. The error is computed as a difference between the estimated and actual values. In the absence of obstacles the robot would dead-reckon to the desired goal location. The velocity and yaw rate computation weighs the output from the "obstacle detection and avoidance" behavior more than the "navigate to goal behavior" in order that the robots stay safe. The "goal generation" behavior is currently implemented as a FIFO queue. Inputs from the mapping behavior are timestampped and queued to a single execution list. The head of this execution list is sent as a goal to the "navigate to goal" behavior. Only one goal is sent to the "navigate to goal" behavior at any time. The decision on when to activate the next goal is made by the "goal generation" behavior, which estimates the degree of progress of the prior goal. Certain events and user inputs can pre-empt the queue. Any input from these channels is treated as a LIFO queue or a stack. This allows the user to interrupt the robot in the midst of a task.

5. RESULTS WITH TEST TASK(S)

The system described here has been tested with two simple tasks to date. The operator initializes the system with a set of GPS coordinates that define the perimeter of interest. The robot locations are registered to a common coordinate frame which is displayed to the operator on the user interface. The first task used for testing was an independent patrolling activity by the two robots. This is shown in Figure 7. The two Pioneers are tasked to patrol the circular areas A and B. The figure shows a trace of the system over time. The cluster of "+" symbols denotes the places that the first Pioneer visited over the duration of the experiment. The cluster of "o" symbols denotes the places that the second Pioneer visited over the duration of the experiment. The two ground robots were not tasked to follow the helicopter or to respond to alarms outside their region of interest. The helicopter is flown by the pilot from region H1 to H2 and then to H3.

The second task we have used for testing involves the automatic retasking of the robots by simulated "alarms" on the area perimeter. The robots navigate to the alarm scene to provide live video while maintaining basic patrolling functionality. When an alarm goes off (alarms are simulated by user keyboard input to the system), the robot nearest

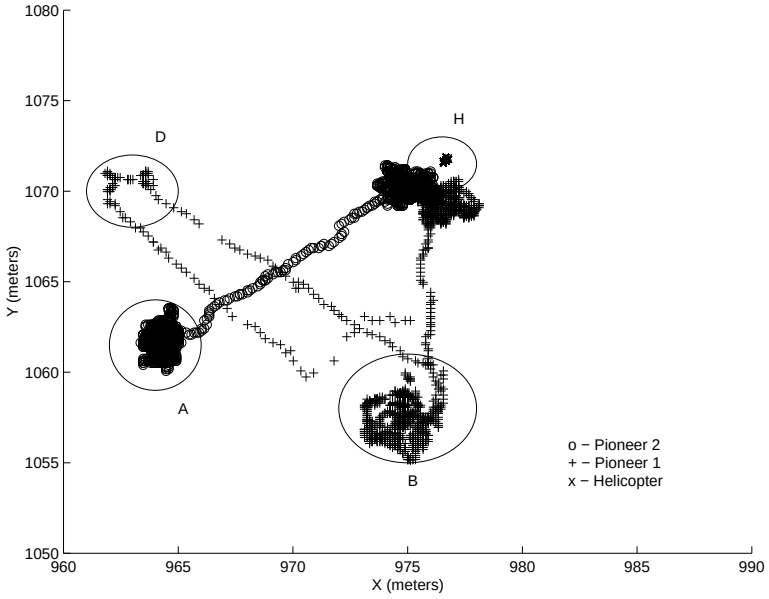

Figure 8. The locations of the three robots over time. The two ground robots are exploring their designated areas of interest (A and B) while responding to an alarm (D) and following the helicopter (H).

to the alarm disengages the patrolling behavior and navigates to the site of the alarm. Since this is a reconnaissance-style scenario, the robot simply returns a close-up view of the scene of the alarm to the operator in the form of a live video feed. The other robot continues patrolling.

A trace of this test is shown in Figure 8. The two Pioneers are initially tasked to follow the helicopter, which is at location H. As the figure shows, the two Pioneers stay within range of the helicopter and maintain a loose formation. The first event is a simulated alarm at location B. One of the Pioneers responds by navigating to location B and patrols there. When a second alarm goes off (at location D) the same robot navigates to region D, sends an image back to the operator and returns to region B and continues patrolling. In the meantime, a third alarm (near region A) causes the other Pioneer (which is still at location H near the helicopter) to patrol area A.

In these tests the helicopter was flown by the human pilot and the Pioneers were autonomous. The only input from the operator is at the start of the task when the perimeter and initial locations are defined.. When the helicopter is fully autonomous we expect to be able to load a flight plan (in the form of several via points) and have the rest of the robots follow in loose formation.

In this short description we have shown examples of the system functionality. A detailed statistical analysis of the behavior of the group[12] is the subject of a forthcoming paper. Videotapes of the current system are available at http://www-robotics.usc.edu/brochure/afv.html.

6. RELATED LITERATURE

For automated full-size helicopter control both model-based[13] (using feedback linearization) and model-free[14,15] (using genetic algorithms and fuzzy logic, respectively) approaches have been used. The two approaches have also been merged[16] by combining neural networks with feedback linearization. Similarly, for smaller radio controlled (RC) model helicopters model-based techniques[17,18] have been applied as have model-free[19,7] (fuzzy logic) techniques. Both methods have also been used in combination.[20] What separates our work from the model-based approaches is its model-free nature and online adaptive capability.

The system described in this paper is designed in the behavior-based philosophy. Behavior-based systems grew out of reactive architectures, which consist of reactive sensor-effector rules that enable real-time system response.[8] While reactive systems are limited to their constituent built-in rules, and thus allow no run-time representations

or adaptation, behavior-based systems have no such limitations. Consisting of collections of behaviors (processes or control laws), they use a distributed approach to representation but still retain real-time properties of reactive systems.[21] As such, they are particularly well suited for group robotics, which require both real-time response and adaptivity, in additional to low-overhead computation that scales well with group size. Since the early 1990's, reactive and behavior-based systems have been used to demonstrate various group behaviors, including flocking,[22] foraging,[22] soccer playing,[23,24] movement in formation,[25,26] planar object pushing,[3,27] and others. While in single-robot control hybrid systems are a popular choice of control architecture,[28,29] practice in the last several years has favored behavior-based systems for group robotics. The move toward heterogeneous group robotics imposes more challenges on the controller than homogeneous systems of the type that have been largely studied to date. As a result, the described system is an example of how a well-structured behavior-based approach can be used to coordinate a multi-robot system consisting of very different robotic agents.

ACKNOWLEDGMENTS

The authors would like to thank several members of the Robotics Research Labs at USC for their help with various aspects of this project. In particular we thank George Bekey, Gerald Bohne, Goksel Dedeoglu, Brian Gerkey, Puneet Goel, Steve Goldberg, Kale Harbick, Francisco J. Mesa-Martinez, Monica Nicolescu, Ken Payne, Richard Vaughan and Barry Werger.

This work is supported in part by Jet Propulsion Labs, California Institute of Technology under contract #959816 and contracts #F04701-97-C-0021 and #DAAE07-98-C-L028 from DARPA.

REFERENCES

1. R. C. Arkin, "Cooperation without communication: Multiagent schema based robot navigation," *Journal of Robotic Systems* **9**, April 1992.
2. T. Fukuda, S. Nadagawa, Y. Kawauchi, and M. Buss, "Structure decision for self organizing robots based on cell structures - cebot," in *IEEE International Conference on Robotics and Automation*, pp. 695–700, IEEE Computer Society Press, Los Alamitos, CA, (Scottsdale, Arizona), 1989.
3. C. R. Kube and H. Zhang, "Collective robotic intelligence," in *From Animals to Animats: International Conference on Simulation of Adaptive Behavior*, pp. 460–468, 1992.
4. M. J. Matarić, "Minimizing complexity in controlling a collection of mobile robots," in *IEEE International Conference on Robotics and Automation*, pp. 830–835, (Nice, France), May 1992.
5. M. J. Matarić, "Interaction and intelligent behavior," Tech. Rep. AI-TR-1495, MIT Artificial Intelligence Lab, 1994.
6. M. J. Matarić, "Issues and approaches in the design of collective autonomous agents," *Robotics and Autonomous Systems* **16**, pp. 321–331, December 1995.
7. J. F. Montgomery, A. H. Fagg, and G. A. Bekey, "The USC AFV-I: A behavior-based entry in the 1994 International Aerial Robotics Competition," *IEEE Expert* **10**, pp. 16–22, April 1995.
8. R. A. Brooks, "A robust layered control system for a mobile robot," *IEEE Journal Robotics and Automation* **2**, pp. 14–23, March 1986.
9. J. F. Montgomery and G. A. Bekey, "Learning helicopter control through 'teaching by showing'," in *1998 IEEE Conference on Decision and Control*, December 1998.
10. G. Dedeoglu, M. Matarić, and G. S. Sukhatme, "Incremental, online map building with a mobile robot," in *Proceedings of the 1999 SPIE Conference*, 1999.
11. P. Goel, S. Roumeliotis, and G. Sukhatme, "Robust localization using relative and absolute position estimates," in *Proceedings of the 1999 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1999.
12. G. S. Sukhatme, J. F. Montgomery, and M. J. Matarić, "A heterogenous robot group for reconnaissance and surveillance." In Preparation.
13. D. Y. Maharaj, *The Application of Nonlinear Control Theory to Robust Helicopter Flight Control*. PhD thesis, Department of Aeronautics, Imperial College of Science, Technology, and Medicine, 1994.
14. C. Phillips, C. L. Karr, and G. Walker, "Helicopter flight control with fuzzy-logic and genetic algorithms," *Engineering Applications of Artificial Intelligence* **9**, pp. 175–184, April 1996.
15. C. Cavalcante, J. Cardoso, J. J. G. Ramos, and O. R. Neves, "Design and tuning of a helicopter fuzzy controller," in *Proceedings of 1995 IEEE International Conference on Fuzzy Systems*, vol. 3, pp. 1549–1554, 1995.

16. B. S. Kim, *Nonlinear Flight Control Using Neural Networks*. PhD thesis, Department of Aerospace Engineering, Georgia Institute of Technology, 1993.
17. O. Amidi, *An Autonomous Vision-Guided Helicopter*. PhD thesis, Department of Electrical and Computer Engineering, Carnegie Mellon University, 1996.
18. P. DeBitetto, E. N. Johnson, E. Lanzilotta, C. Trott, and M. Bosse, "The 1996 MIT/Boston University/Draper Laboratory autonomous helicopter system," in *Proceedings of 1996 Association for Unmanned Vehicle Systems 23rd Annual Technical Symposium and Exhibition*, pp. 911–920, (Orlando, FL), 1996.
19. M. Sugeno, M. F. Griffin, and A. Bastian, "Fuzzy hierarchical control of an unmanned helicopter," in *17th IFSA World Congress*, pp. 179–182, 1993.
20. F. Hoffman, "Soft computing techniques for the design of intelligent systems." OAI Workshop on Fuzzy Logic (http://HTTP.CS.Berkeley.EDU/~fhoffman/oai97/oai97.html), February 1997.
21. M. J. Matarić, "Behavior-based control: Examples from navigation, learning, and group behavior," *Journal of Experimental and Theoretical Artificial Intelligence* **9**(2–3), pp. 323–336, 1997.
22. M. J. Matarić, "Designing and understanding adaptive group behavior," *Adaptive Behavior* **4**, pp. 50–81, December 1995.
23. M. Asada, E. Uchibe, S. Noda, S. Tawaratsumida, and K. Hosoda, "Coordination of multiple behaviors acquired by a a vision-based reinforcement learning," in *Proceedings, IEEE/RSJ/GI International Conference on Intelligent Robots and Systems*, (Munich, Germany), September 1994.
24. B. B. Werger, "Principles of minimal control for comprehensive team behavior," in *Proceedings of ICRA-98*, pp. 3504–3509, (Leuven, Belgium), 1998.
25. L. E. Parker, "A performance-based architecture for heterogeneous, situated agent cooperation," in *AAAI-92 Workshop on Cooperation Among Heterogeneous Intelligent Systems*, July 1992.
26. T. Balch and R. Arkin, "Behavior-based formation control for multi-robot teams," *IEEE Transactions on Robotics and Automation* **to appear**, 1999.
27. M. J. Matarić, M. Nilsson, and K. T. Simsarian, "Cooperative multi–robot box–pushing," in *Proceedings, IROS-95*, IEEE Computer Society Press, (Los Alamitos, CA), 1995.
28. J. H. Connell, "Sss: A hybrid architecture applied to robot navigation," in *IEEE International Conference on Robotics and Automation*, pp. 2719–2724, (Nice, France), 1991.
29. E. Gat, "On three-layer architectures," in *Artificial Intelligence and Mobile Robotics*, D. Kortenkamp, R. P. Bonnasso, and R. Murphy, eds., AAAI Press, 1998.