Decentralized Cooperative Control

A multivehicle platform for research in networked embedded systems

Daniel Cruz, James McClintock, Brent Perteet,

Omar Orqueda, Yuan Cao, and Rafael Fierro

Recent advances in communication, computation, and embedded technologies support the development of cooperative multivehicle systems [1]. For the purposes of this article, we adopt the following definition of cooperative behavior [2]: "Given some task specified by a designer, a multiple-robot system displays cooperative behavior if, due to some underlying mechanism (i.e., the 'mechanism of cooperation'), there is an increase in the total utility of the system". The development of cooperative multivehicle systems is motivated by the recognition that, by distributing computer power and other resources, teams of mobile agents can perform many tasks more efficiently and robustly than an individual robot. For example, teams of robots can complete tasks such as multi-point surveillance, distributed localization and mapping, and cooperative transport.

To facilitate the development of cooperative control systems and mobile sensor networks, we have created a COoperative MultivehiclE Testbed (COMET) for research and experimentation. The objective of this platform is to implement and validate new cooperative control techniques. This article provides an overview of COMET. Specifically, we discuss the hardware and software components of the platform and show how the components are integrated to maximize flexibility and expandability. We provide examples to illustrate the types of applications that can be explored using this platform.

COMET consists of a team of ten mobile sensor agents as shown in Figure 1. Each agent is equipped with an onboard embedded computer that interfaces with sensors and actuators on the robot. This onboard computer supports wireless network access so that the robots can be queried and controlled

from the Internet or a local area network. In addition, we have developed models for simulating the rigid body dynamics on an open-source simulator. The availability of a tightly coupled simulation environment allows researchers and developers to preview the functionality of their algorithms, which helps to minimize testing and development time.

COMET is designed to be scalable, accommodating experiments with an arbitrary number of vehicles. Depending on the application, new sensors can be added on-the-fly to the sensor pool available for each agent. As we discuss later, each vehicle is itself a network of sensors. Thus, COMET is a collection of small, dynamic networks. This architecture enables COMET to serve as a testbed for studying decentralized, dynamic, real-time sensor networks as well as networked control and cooperative control algorithms. Additionally, the platform offers a unique educational opportunity. While working with these robots, students gain first-hand experience with technologies such as networked control systems, embedded computing systems, and controller area networks (CAN). Videos of COMET performing several of the cooperative control tasks described in this article are available online [3].

The remainder of this article is organized as follows. In the related work section, we discuss similar work being done at other universities and research facilities. The second section presents a vehicle description, discussing the main hardware and embedded sensing components of the platform. Next, the software architecture, vision library, and graphical user interface (GUI) are presented. In the remaining sections, we discuss several control techniques that are implemented on the platform. We begin by showing a suite of basic control behaviors, such as obstacle avoidance. Next, control techniques that are more advanced are described. In the final section, we draw conclusions and suggest future work.

Related Multivehicle Platforms

In the last few years, multivehicle platforms have been developed at several universities and research labs. Some of the platforms involve hovercraft vehicles such as the Caltech multivehicle wireless testbed

[4] and the University of Illinois HoTDeC [5]. Other testbeds feature unmanned ground vehicles and unmanned aerial vehicles, such as MIT's multivehicle testbed [6] and the University of Pennsylvania's Multiple Autonomous Robots (MARS) testbed [7], [8]. The MARS platform uses ground vehicles based on the Tamiya TXT-1 chassis, which we use in COMET. Cornell University's RoboFlag testbed [9] includes several small robots with local control loops that work cooperatively to achieve a common goal. These robots, designed for indoor experiments, use an overhead camera for localization. Each vehicle carries a simple microcontroller unit, which is in charge of commanding the actuators on the robot. High-level control is accomplished using a remote workstation. Researchers at Brigham Young University use a multivehicle testbed [10] for cooperative robotics research in their MAGICC lab. Their initial system consisted of five nonholonomic robots, with an onboard Pentium-based computer and wireless communications.

In this article, we discuss several coordinated control algorithms that are implemented on COMET. Our approach is to design low-level robotic behaviors and combine them to generate high-level controllers, as suggested in [11]. More recently, several authors have approached the problem of behavior-based control using techniques from hybrid systems. For example, [12] provides an overview of multimodal behavior-based control with an emphasis on using low-level behaviors to describe a high-level task as compactly as possible. The behaviors that we use are based on potential field control (PFC), a technique used extensively by robotics researchers [13]. Although most of the work focuses on fully actuated planar robots, some results discuss the more difficult problem of using PFC for nonholonomic robots. For example, [14] approaches this problem by generating a trajectory for a holonomic robot and then using nonholonomic trajectory tracking.

Although our platform has some similarities with those listed above, COMET provides a combination of features that cannot be found elsewhere. For example, to facilitate experimental use, COMET supports online sensor reconfiguration. We use PC-104 computers on the robots to maximize

compactness and modularity. Most of our onboard sensors communicate with the computers using a locally designed CANBot board [15]. Additionally, our platform offers a system for vision-based multivehicle coordination by combining inexpensive webcams with a NameTag system. Finally, we make extensive use of open-source software, including Linux and Player [16].

Vehicle Description

COMET consists of ten robots based on the TXT-1, a one-tenth scale R/C monster truck from Tamiya, Inc. Each truck is modified to be part of an adaptable team of networked mobile agents. The vehicles are small enough to operate in a laboratory environment yet sturdy enough to negotiate rough outdoor terrain. The robot can support a load of 3.7 kg, allowing it to carry an assortment of computers, sensors, and other objects. A list of COMET's major components and their respective vendors is provided in the sidebar "COMET Components and Vendors."

The system allows users to configure each vehicle with various sensors depending on the application requirements. Figure 2(a) illustrates the sensors that are currently available for use. The mechanical specifications of the platform are shown in Table I.

The controller area network is used to communicate with onboard sensors because of its low latency time and high speed (up to 1 Mbps) [17]. Each device or sensor attached to the vehicle is accompanied by a small board as a communication interface to the CANBus. These boards use PIC18F series microcontrollers to process sensor data. Since CAN uses a message-based protocol, addresses are not required. The CANBus system allows sensors and actuators to be added and removed from the system without reconfiguring the high-level control structure. When a module is connected to the bus, the low-level software configures the device to make its information available to the higher level software. This plug-and-play capability allows large sensor networks to be built without the need to reconfigure the entire system.

Mechanical Modifications

To convert an off-the-shelf Tamiya R/C truck into a mobile robot for research in networked embedded systems, several modifications are required. The remainder of this section describes the changes and additions that we make to each vehicle.

To provide odometry for each robot, the front wheels, which can rotate independently, are equipped with separate 400-counts-per-revolution quadrature optical encoders. By using two encoders, the vehicle's translational and rotational speeds can be calculated. The encoder boards and code wheels are installed as shown in Figure 3.

Since the plastic truck body that comes with the TXT-1 is not conducive to mounting sensors, a precision-milled aluminum plate is attached to each robot as shown in Figure 4. This plate is designed to be light weight but sturdy enough to support computers and sensors.

Finally, the suspension system on the chassis is improved through the addition of one shock absorber per wheel. This upgrade increases the maximum load that the vehicle can carry without leaning while raising the chassis slightly higher off the ground.

The CANBot Control Board

The CANBot control board is designed to provide a simple interface between sensors connected to the CANBus and the vehicle's onboard computer. A functional block diagram of this board is shown in Figure 2(b). The three main features of this board include device management, vehicle control, and pulse-width-modulated (PWM) signal generation. These features are described in more detail below.

Device Management

The CANBot board performs active bus monitoring to relieve the high-level controller of this task. The device manager broadcasts a data packet with information about which devices are connected and active at any moment in the CANBus. When a device stops transmitting information, the device manager attempts to reset the device in order to bring it back into operation. If no response is received, the device manager signals the high-level controller that the sensor is no longer present.

Vehicle Control

The CANBot control unit provides the high-level controller with a simple interface for adjusting the robot's translational and rotational velocities. To complete this task, the unit uses two PID controllers that run on a single microcontroller. More details on speed control are provided in the following subsection.

Pulse-Width-Modulated Signal Generation

PWM signal generation is an important capability of the CANBot control unit, since the servo motors use PWM for input commands. All of the servos on the platform, including the motor drive, steering servo, and camera pan servo, are controlled using commands sent over the CANBus. The CANBot board can simultaneously control eight servos through its PWM-generation chip. A typical servo connected to this unit has a resolution of 0.043 deg per control bit.

Optical encoders are extremely insensitive to noise. However, electrical noise on the signal lines can degrade the accuracy of the odometry system, for instance, by adding counts. Since PWM uses high-frequency signals, care must be taken to avoid coupling noise into sensitive systems. Standard noise-reduction practices are followed when designing the CANBot control board and wiring the signal lines from the board to the servos.

Vehicle Kinematics and PID Tuning

Considering the monster truck chassis, a kinematic description of COMET is given by the car-like model [18]

$$\dot{x}_i = v_i \cos\theta_i, \tag{1}$$

$$\dot{y}_i = v_i \sin\theta_i, \tag{2}$$

$$\dot{\theta}_i = \frac{v_i}{l} \tan\phi_i, \tag{3}$$

where $(x_i,\ y_i)$ and θ_i are the position and orientation of robot i; v_i and ϕ_i represent the robot's translational velocity and front wheel steering angle, respectively; and l is the distance between the robot's front and rear wheels. Note that (3) can simply be written as

$$\dot{\theta}_i = \omega_i, \tag{4}$$

where ω_i is the angular velocity of the vehicle.

To facilitate high-level control design, COMET's low-level speed controller is set up to accept velocity commands $u_i^c = [v_i^c \quad \omega_i^c]^T$. The CANBot control board provides this feature by using a PID controller to maintain the robot's translational and rotational speeds at desired values. This low-level controller guarantees that, after a short period of time,

$$||u_i^c - u_i|| < \varepsilon,$$

where $u_i = [v_i \quad \omega_i]^T$ is the i^{th} robot's speed vector and ε is a small positive constant. The controller requires measured values of the translational speed v_i and rotational speed ω_i as feedback. These speeds are estimated by the CANBot control unit using the front wheel encoders. Encoder counts are read from the CANBus using a sampling time $\Delta T = 20$ ms. Each time the encoders are read, the previous

count is subtracted from the current count to determine the number of ticks that occurred in that 20-ms interval. The counter rolls over to zero once it reaches about 64,000 ticks; however, the subtraction routine automatically accounts for turnover unless the truck travels more than about 10 m in any 20-ms interval, which is impossible. Because the encoders have 400 lines and are quadrature, their effective resolution is 1600 ticks per revolution, that is, 0.225 deg. Using this value together with the wheel diameter of 6 in, the encoder distance factor is approximately $\zeta = 0.3$ mm/tick. This calculation has been verified experimentally on flat, even terrain. Since random wheel slippage can occur [19], the odometry is calibrated to determine an average distance factor for a given type of surface. Using the distance factor together with the vehicle width $W = 37.5$ cm, the translational velocity v_i and rotational velocity ω_i are estimated as

$$v_i = \frac{(d_R + d_L)\zeta}{2\Delta T}, \tag{5}$$

$$\omega_i = \frac{(d_R - d_L)\zeta}{W\Delta T}, \tag{6}$$

where d_R and d_L are the number of ticks in one sampling interval by the robot's right and left encoders, respectively.

To maintain the robot's translational and rotational velocities, the CANBot board uses a PID controller of the form

$$V_i(n) = K_p e_i(n) + K_i \sum_{k=0}^{n} e_i(k) + K_d[e_i(n) - e_i(n-1)],$$

where $V_i(n) = \begin{bmatrix} V_i^{Drive} & V_i^{Steer} \end{bmatrix}^T$ are the PID controller outputs that drive the robot's motors and steering servos, respectively, $e_i(n) = [v_i^c - v_i \quad \omega_i^c - \omega_i]^T$ is the difference between the commanded and measured translational and rotational speeds, and $K_p = \mathrm{diag}(K_{pv}, K_{p\omega})$, $K_i = \mathrm{diag}(K_{iv}, K_{i\omega})$, and $K_d = \mathrm{diag}(K_{dv}, K_{d\omega})$ are controller gains. This PID controller runs on a PIC microcontroller. To calculate the error signal, the microcontroller estimates v_i and ω_i using (5) and (6). The commanded translational

speed v_i^c and rotational speed ω_i^c are provided by the high-level control program running on the PC-104. The values $V_i^{Drive} \in [-2048, 2047]$ and $V_i^{Steer} \in [-2048, 2047]$ are discrete controller outputs. After the PID controller executes, V_i^{Drive} is converted to a PWM signal, which is sent to the dc motor drive. V_i^{Steer} is also converted to a PWM signal, which controls the steering servo.

The controller gains K_d, K_i, and K_p are hand tuned using the dynamic model

$$\begin{aligned} m_i \dot{v}_i &= -\eta v_i + K_1 V_i^{Drive}, \\ J_i \dot{\omega}_i &= -\sigma \omega_i + K_2 V_i^{Drive} V_i^{Steer}, \end{aligned}$$

where m_i is the mass of the i^{th} robot, J_i is the robot's moment of inertia, η and σ are damping factors, K_1 is a constant with units of force, and K_2 is a constant with units of torque. Since these values are highly dependent on the motors and servos, they are not listed here. This dynamic model is used for PID tuning because it accounts for the coupling between the translational and rotational speeds. Once the performance of the controller is satisfactory, the controller gains are set in the CANBot firmware. The gains that we use are

$$\begin{aligned} &K_{pv} = 0.80 \text{ s/m}, \quad && K_{dv} = 1.00 \text{ s/m}, \quad && K_{iv} = 0.02 \text{ s/m}, \\ &K_{p\omega} = 0.40 \text{ s}, && K_{d\omega} = 0.01 \text{ s}, && K_{i\omega} = 0.10 \text{ s}. \end{aligned}$$

Figure 5 shows the controller performance on even terrain.

Onboard Computing

An Advanced Digital-Logic PC-104 computer hosts the local control software on each vehicle. The PC-104 system, which has the same capabilities as a desktop system, has a reduced size that fits on COMET. The PC-104 is equipped with a 40-GB hard drive, FireWire, Kvaser CAN card, WiFi support, and a Pentium M processor. The system features the RedHat Fedora operating system and runs the Linux

kernel version 2.6. The onboard computer, which runs high-level control tasks, interfaces with the vehicle through the CANBot Control Unit. The CANBot Control Unit is the only CAN device that is mandatory for each vehicle.

Sensor Suite

This section describes the suite of sensors available for use on the platform. Each sensor features CAN connectivity based on the PIC18F258/248 microcontroller.

Infrared Sensors

The trucks carry infrared (IR) arrays based on the Sharp 2Y0A02 infrared long-distance measuring sensor. Each Sharp sensor outputs an analog voltage proportional to object distance over a range of 0.2 m to 1.5 m. The IR array consists of three IR sensors mounted at a 45-deg offset. An IR/CAN control-board interface with the IR array transmits the measured distances through the CANBus at a 25-Hz sampling rate.

Global Positioning System

The global positioning system (GPS)/CAN control board can support any GPS receiver that outputs standard NMEA 0183 sentences. A Garmin GPS 18 receiver (the 5-Hz version) is used on the platform. This device, which is accessed using RS-232, provides data on longitude, latitude, altitude, heading, speed, satellites detected, and dilution of precision (which indicates GPS error based on the positions of the satellites relative to the GPS receiver). The GPS features wide area augmentation system technology to enhance its accuracy using a network of ground stations and satellites. When this feature is enabled, the position accuracy is approximately 3 m. The sensor has an adjustable update rate of up to 5 Hz.

When a GPS receiver is connected to a vehicle, its position data are considered as truth. If present, the inertial measurement unit (discussed below) and encoders are used as part of the positioning system to provide higher resolution over short distances. However, the output of the positioning system is reset to the GPS position if the difference between odometry measurements and the GPS position is greater than the GPS resolution.

Inertial Measurement Unit

The inertial measurement unit (IMU) is the Microstrain 3DM-G. This IMU provides temperature-compensated triaxial orientation information in pitch (θ), roll (ϕ), and yaw (ψ) by integrating data from 9 sensors (3 gyros, 3 accelerometers, and 3 magnetometers) and measuring against a local coordinate system. This information is updated at a rate of 75 Hz. Gyro-enhanced odometry is more accurate than encoder-based odometry, particularly in determining the heading of the vehicle, since this angle is provided by the IMU instead of differences in wheel rotation.

While 2D gyro-enhanced odometry is automatically provided when the IMU is connected to the system, the user can alternatively request 3D position, which makes more comprehensive use of the sensor. The interface board, which samples data from the IMU using the RS-232 data port, has the same design and layout as the GPS interface board. The only difference between the two is the firmware loaded into the microcontroller's flash memory. Since both the GPS and IMU operate by reading low-power electromagnetic signals, their performance is affected by electromagnetic interference produced by the vehicle itself and the environment. For this reason, the sensors are mounted as far away from the onboard computer as possible. Also, large metal objects in the environment or onboard can degrade the accuracy of the magnetometers.

Laser Range Finder

The most recent addition to COMET's sensor suite is a Hokuyo URG-04LX laser range finder (LRF). Mounted on top of the PC-104 enclosure, the LRF is connected to the computer by USB. The device uses a spinning laser coupled with advanced detection circuitry to create a 2D map of the distance to nearby objects. This sensor has a range of 5 m and offers a wide viewing angle of 240 deg. The LRF has a scanning rate of 10 Hz, an angular resolution of 0.36 deg, and accuracy of ± 10 mm, which makes it an excellent tool for autonomous vehicles exploring an unknown environment. Experimental applications of a LRF include obstacle avoidance and contour following.

Vision

A video camera is available for use on the robots. Due to the large volume of data generated by the camera, the sensor does not connect to the CAN network. Instead, we use a FireWire camera connected directly to the onboard computer. Currently, our vision library works with BumbleBee stereo cameras from Point Grey as well as Fire-i digital cameras from Unibrain [20]. More details about the vision library are provided below.

Power System

The vehicle electronics, including the onboard computer and all of the sensors, are powered by a Powerizer 11.1-V, 5500-mAh, lithium ion battery. Lithium ion batteries are light weight, storing more energy per unit mass (110-160 W-h/kg) than batteries using heavier metals such as nickel metal hydride (30-80 W-h/kg). On a full charge, the Powerizer battery can keep the electronics running for approximately 3 hours. Additionally, a Datel UHE-5/5000 DC/DC converter provides the 5-V input required by several of the components.

A second battery is used for the motors because the current draw surges when the motors spin up causing voltage fluctuations, which are problematic for the sensitive electronics. Also, the lithium ion battery cannot supply the current needed by the motors. Instead, we use a 6-cell, 7.2-V, 3300-mAh, nickel metal hydride battery. This battery connects directly to the speed controller and, depending on usage, lasts for several hours on a charge.

Vehicle Operation and Maintenance

Overall, the robots are robust, requiring little maintenance for everyday operation. For attaching and reconfiguring sensors, each robot's aluminum plate is drilled and tapped with approximately 100 mounting holes. An aluminum case houses the onboard computer to provide protection, with a removable cover for easy access. Everyday maintenance of the robot includes cleaning after outdoor use and recharging motor and computer batteries.

To keep the vehicles in working order, some routine maintenance is required. For example, the batteries are replaced as they wear out, usually after about 50 chargings. Also, the dampers in the suspension system lose a small amount of oil over time so they must be checked and refilled periodically. On rare occasions mechanical components such as the lower plastic supports for the shield that holds the steering servos, as well as the gears inside the servos, are damaged and must be replaced.

TXT Software Libraries

In this section we elaborate on the higher level software developed for COMET (see Figure 6). This software is written using C++ and Playerlib to interface with Player. This section describes the Player driver that interfaces with the in-vehicle CAN network, the vision library, and the control functions available to developers.

The Player/Stage/Gazebo Project

The Player/Stage/Gazebo [16] project is an ongoing open source software development effort that offers a suite of tools for research in robotics. The source code, which is available for free download from *sourceforge.net*, is protected by the GNU General Public License. Since the source code is available for download and is open to modifications, researchers and developers often contribute drivers for new sensors and mobile robots. The software can run on several platforms such as Solaris, Linux, and Mac OSX.

The project is comprised of three components that can be used individually or together. *Player* is an application that provides a network interface for sensors and actuators on the robot. Additionally, since Player allows multiple connections, more than one client can access each robot's resources at the same time. This feature is useful in multivehicle control and coordination applications. Player supports hardware such as LRFs, cameras, GPSs, and most MobileRobots (previously ActivMedia) robots. *Stage* and *Gazebo* are 2D and 3D world simulators, respectively. Both of these applications can simulate populations of agents, sensors, objects, and environments.

Each vehicle executes Player using its onboard PC-104 computer. Once Player is running, the vehicle is controllable through the network. A control program might run on the local machine or on a remote computer. Since COMET uses several communication networks, systematic and non-systematic delays can affect the performance of the controllers. The lower level controller deals with the CANBus network delays and latency. These delays are generally insignificant and can be disregarded. The higher level control assumes information is readily available at each sampling interval. However, when the controller operates using the local area network, delays can be significant and affect the performance of the system [21]. We plan to model these delays and their effects on performance of the testbed in future work.

A Gazebo model that supports the same sensors as our trucks is used to simulate COMET. Since Gazebo has a standard Player network interface, high-level control programs can connect to virtual robots to read sensor values and execute control commands. In fact, from the standpoint of a high-level control program, virtual and real robots are identical. Without any changes, a program that can run on the PC-104 to control a real robot can run on a desktop and control a virtual robot in Gazebo. An example of a Gazebo simulation is shown in Figure 7(a).

The TXT CAN Player Driver

The TXT Player driver is a bridge between Player and the CAN sensor network. A PCMCIA-CAN card or USB-CAN connector is used as the interfacing hardware, which is facilitated by the Kvaser CANLib library of functions. The currently supported CAN-enabled devices include the Garmin GPS, CANBot control unit, MicroStrain IMU, and IR distance sensors. Player includes the *camera1394* driver to interface with FireWire cameras. The TXT Player driver updates information from every sensor connected to the CANBus at 1 kHz. This speed exceeds the update rates of our current sensors, and thus can facilitate integration of faster sensors in the future.

Mobile Robot Vision

When a group of mobile agents attempts to perform a coordinated behavior such as flocking [22], leader-following [23], or a cooperative operation [24], each agent in the group must identify its neighbor agents and estimate their positions and orientations. One way to obtain this information is through an onboard vision sensor [25], [26]. To accomplish robot identification and localization, COMET uses markers or visual tags arranged on the back of each robot on a 3D-truncated octagonal-shaped structure, as shown in Figure 4. Each face of this visual tag has a code that provides the vehicle's ID as well as the position of the face in the 3D-visual marker. This information allows a vision sensor to identify the vehicle and

estimate its pose relative to the sensor coordinate system. The ink pattern is black and white to reduce lighting and camera sensitivities.

The vision library analyzes an image using a five-step process. During *video capture and transformation*, images from the camera are captured and converted into OpenCV's RGB format. The second step is *filtering and thresholding*, in which the image is converted to grayscale and thresholding based on a modified Otsu method [27] is performed. In the third step, a *point selection/face labeling* algorithm is applied. A search and identification of each face marker in the binary image is performed. To this end, all of the contours on the binary image are extracted, and the rectangles enclosing each marker are identified. After estimating the four corners of each rectangle, a search for markers is performed. If a valid marker is found, the corresponding four corners are stored for pose estimation. The fourth step is *pose estimation*, which consists of a nonlinear optimization problem [28]. To reduce the likelihood of false local minima, the optimization problem is initialized with the POSIT algorithm [29]. The position of the leader robot with respect to the camera position as well as the relative rotation of the leader robot with respect to the camera plane are estimated. The fifth step involves the *diffeomorphic transformation* of the measured values to the desired ones. More details about COMET's vision library are given in [20]. Figure 8 shows the steps involved in the NameTag detection processes.

Robot Control Software

The Graphical User Interface

To coordinate the multivehicle network, we have created a GUI for use by a human operator. The interface is written in C++ using the Trolltech Qt application framework. The GUI can run on any platform that Qt supports, such as Windows, Linux, and Mac. The GUI is designed to support TCP/IP communication and interface with Player to command and monitor the mobile agents. In a laboratory environment, the robots can connect to a wireless access point with Internet access, and the GUI can

then be run on any computer connected to the Internet. To accommodate field work, an ad hoc wireless network is set up, and the GUI is run on a laptop with wireless capability. The GUI is shown in Figure 9.

Methods of Operation

The GUI has several features that allow users to configure COMET's level of autonomy. For example, users can choose between individual and team control. At the lowest level, users can command individual robots by setting translational and rotational velocities for each vehicle. Alternatively, the GUI can be used to specify a set of waypoints for each vehicle, thus allowing the agent to plan its trajectory using a PFC [30]. Finally, to control team maneuvers, the GUI provides a set of basic building block functions. Specifically, teams can move in a leader-follower or flocking formation to a desired location. The team can also be commanded to travel along a route specified by waypoints. More details about these coordinated control behaviors are provided below.

The GUI also facilitates centralized or decentralized coordination of the team. In centralized coordination mode, the control algorithms for each vehicle, for example, obstacle avoidance, go-to-goal, or waypoint navigation, run on the same computer as the GUI. This mode requires a powerful computer since a copy of the navigation algorithm must be run for each vehicle. In decentralized coordination mode, the GUI is used only to issue high-level commands and monitor the state of the multivehicle system. Each vehicle computes and executes its own controller based on the mission at hand. Navigation and obstacle avoidance are also executed locally on the agent's onboard computer. Instead of sending a large amount of real-time control data, the GUI communicates only waypoints to the vehicles.

Finally, the GUI provides the user with advanced monitoring capabilities. When a map of the environment is supplied, the interface shows the location of each robot as well as the robot's proximity to obstacles. When a robot moves in the real world, the interface shows a moving icon on the map. The GUI

can display more detailed information about any individual agent, including its position, orientation, and velocity. To support reconnaissance missions, live video from any agent equipped with a camera can be viewed. Users can pan the camera to view objects that are not directly in front of the vehicle. However, the video transfer makes heavy use of the network.

Levels of Autonomy

As mentioned above, the level of autonomy of COMET depends on how the team is configured by the GUI to operate. For example, to run in centralized coordination mode, the robots must stay in contact with the GUI. In decentralized mode, however, the GUI might disconnect after uploading a mission to the robots. Also, when team maneuvers are requested, the robots exercise greater autonomy than when the user specifies waypoints for each vehicle. The tradeoffs relate to the amount of user input and scope of operation. When the robots act autonomously, the user has less finely tuned control over their actions, although entering control commands is less of a burden. On the other hand, commanding each robot is time consuming but allows for precise control. Also, when the robots act autonomously, they can carry out missions in areas where communication is limited. The major disadvantage is that the user loses the advanced monitoring capabilities of the GUI.

Basic Control Behaviors

This section describes basic control behaviors that are implemented on COMET. These behaviors can be combined in parallel or sequentially to create more complex swarming behaviors. The algorithms are implemented with C++ control functions and run in Linux on the PC-104 computers. Player is used to interface with the vehicle's hardware. Thus, the behavior implementations are subject to the networking delays and sampling time restrictions of the CANBus as well as the sensors and actuators described above.

Go-To-Goal

The go-to-goal behavior is based on a PFC algorithm. PFCs use an attractive potential, which allows the agents to move to a specific location. A PFC can be used for feature tracking, leader-following, waypoint tracking, or any other application in which a vehicle needs to reach a designated goal. The artificial potential $\mathbf{P_a}(x_i, y_i)$ is given by [30]

$$\mathbf{P_a}(x_i, y_i) = \frac{1}{2}\varepsilon[(x_i - x_g)^2 + (y_i - y_g)^2],$$

where (x_i, y_i) is the position of agent i, ε is a positive constant, and (x_g, y_g) is the position of the goal. The attractive force, $\mathbf{F_a}(x_i, y_i)$, can be expressed as

$$\mathbf{F_a}(x_i, y_i) = - \bigtriangledown \mathbf{P_a}(x_i, y_i) = - \begin{bmatrix} \frac{\partial \mathbf{P_a}}{\partial x_i} \\ \\ \frac{\partial \mathbf{P_a}}{\partial y_i} \end{bmatrix},$$

which yields

$$\mathbf{F_a}(x_i, y_i) = \varepsilon \begin{bmatrix} x_g - x_i \\ \\ y_g - y_i \end{bmatrix} \triangleq \begin{bmatrix} F_{a,x_i} \\ \\ F_{a,y_i} \end{bmatrix}. \tag{7}$$

Based on the two values calculated in (7), the desired orientation angle

$$\theta_{i,d} = \arctan2(F_{a,y_i}, F_{a,x_i}) \tag{8}$$

can be determined, where $\arctan2(\cdot)$ in (8) represents the four-quadrant inverse tangent of F_{a,x_i} and F_{a,y_i}. This function uses the signs of these two values to remove the two-solution ambiguity normally associated with the inverse tangent of $F_{a,y_i}/F_{a,x_i}$.

After the orientation relative to the goal is calculated, the proportional controller

$$\omega_i = \pm k \ (\theta_{i,d} - \theta_i),$$

where $k = \frac{\omega_{max}}{2\pi}$ s^{-1} and $\omega_{max} = 0.4$ rad/s, is used to align the vehicle with the desired heading. The translational speed of the vehicle while executing the PFC may vary. In some applications such as feature detection, this speed can be held constant. In other applications such as goal seeking, the speed of the vehicle depends on the distance to the goal. In leader-follower applications, the speed of the leader is used as a feedforward term to increase system robustness.

This behavior has several limitations that must be considered. When using the PFC together with a nonholonomic vehicle, the robot may begin to circle around its goal position. This behavior occurs because noise in the odometry can cause the robot's position estimate to shift laterally, and thus the robot must circle around to correct the error. This problem can be solved by including a threshold in the potential P_a such that the robot does not move when P_a is below a certain value. Additionally, since the go-to-goal behavior does not consider obstructions in the environment, collisions can occur. Go-to-goal behavior can be coupled with obstacle avoidance to overcome this problem as described in the following subsections.

Obstacle Avoidance

Obstacle avoidance (OA) is a basic building block of autonomous robotic systems [31]. The OA behavior can be seen as part of a hierarchical hybrid system [32], which combines both continuous and discrete states. A common technique is to define obstacle avoidance as a discrete control state. When the robot detects an object that is closer than a threshold value, the obstacle avoidance state is used. The hybrid automaton shown in Figure 10 illustrates this idea. Obstacle avoidance makes operating the robot safer by using sonar or IR sensors to detect and avoid obstacles. In our implementation of OA, the agent uses two CAN-enabled IR arrays located at the front of the vehicle body. Since each CAN-enabled IR array has three IR sensors, six IR sensors are available. These sensors are numbered from 0 to 5 as shown in Figure 11. Much like the PFC, this algorithm begins by computing a set of virtual forces. In this case,

the virtual right and left forces are repulsive and are calculated based on four IR sensor readings as

$$F_r = \sum_{n=1}^{2} \frac{\lambda_n}{d_n},$$
$$F_l = \sum_{n=3}^{4} \frac{\lambda_n}{d_n},$$

where d_n is the distance to the obstacle reported by the n^{th} sensor and λ_n is a scaling factor applied to each IR sensor's distance measurement. In other words, λ_n determines the weight of each sensor in controlling the robot. After extensive experimental testing, we chose the value $\lambda_n = [0.7, 0.9, 1.0, 0.7]$, which weighs the repulsive force in favor of objects directly in front of the robot.

The OA controller is a hybrid controller with two discrete states. The algorithm tracks the contour of an obstacle if the obstacle is in the way of the designated goal, but not directly in front of the vehicle. In this contour tracking state, OA uses the proportional controller

$$\omega_i^c = \begin{cases} k_p(d_5 - D_{safe}), & \text{if the obstacle is on the left,} \\ k_p(D_{safe} - d_0), & \text{if the obstacle is on the right,} \end{cases}$$

where k_p is constant gain and D_{safe} is the constant distance at which the perimeter of the obstacle is tracked. This distance is set to 0.2 m. The contour tracking speed is $v_{OA} = 0.6$ m/s. When an obstacle is directly in front and must be avoided, the OA controller enters a collision-avoidance state and calculates the translational and rotational velocities as

$$v_i^c = v_{OA} - k_{vp} \sum_{n=1}^{4} \frac{\lambda_n}{d_n},$$
$$\omega_i^c = k_{\omega p}(F_l - F_r),$$

where k_{vp} and $k_{\omega p}$ are control constants, and v_{OA} is a reduced contour tracking velocity. Once the obstacle is no longer directly in front, the controller switches to contour tracking, and the obstacle's perimeter is tracked as before. The vehicle continues tracking the contour of the obstacle until the obstacle no longer

obstructs the path to the goal. This condition is verified when the distance to the obstacle reported by sensors 1 or 4 (see Figure 11) are $d_1 \geq \rho$ or $d_4 \geq \rho$, where $\rho = 0.25$ m. Once the obstacle is negotiated, the higher level control continues operation.

Goal seeking

Goal seeking emerges from a combination of obstacle avoidance and go-to-goal behaviors using the hybrid architecture shown in Figure 10. The objective of this emergent behavior is to drive the agent to a designated waypoint without colliding with obstacles. This behavior is useful since it enables the vehicle to travel autonomously. Goal seeking is included as a fundamental block of the GUI. Figure 12 shows a mobile agent at its initial position in a goal-seeking experiment along with the trajectory that the agent follows while traveling to its goal.

Figure 13(a) shows the goal-seeking controller's performance. Since the controller is hybrid, it switches between the obstacle-avoidance states discussed above based on the proximity of obstacles in the environment. Figure 13(b) depicts the transition of the controller between the PFC, collision avoidance, and obstacle contour-tracking states. The goal-seeking algorithm is not a path planner and therefore cannot guarantee that the goal is reachable. This behavior is intended to be a basic building block in a library of more complex path-planning algorithms.

Motion-Coordination Algorithms

In this section, we discuss several motion-coordination algorithms used on COMET. Some mathematical preliminaries are required. A multivehicle system can be considered as a dynamic network in which each node represents a mobile agent (vehicle, mobile sensor, robot) with communication, sensing, and control capabilities. We consider a dynamic network Σ composed of N mobile agents that share a configuration space $\mathcal{Q}$. Specifically, following [33], [34], a *dynamic network* Σ of mobile agents is

a tuple $(\mathcal{I}, \mathcal{A}, \mathcal{G}_c, \mathcal{G}_s, \mathcal{H})$, where $\mathcal{I} = \{1, \ldots, N\}$ is the set of unique identifiers representing agents in the network, $\mathcal{A} = \{(X_i, U_i, X_0, f_i)\}_{i \in \mathcal{I}}$ is a set of control systems (physical agents), $\mathcal{G}_c = \{\mathcal{V}, \mathcal{E}_c\}$ is an undirected communication graph, where $\mathcal{V}$ is the set of nodes and $\mathcal{E}_c$ is the communication edge map, and $\mathcal{G}_s = \{\mathcal{V}, \mathcal{E}_s\}$ is a directed sensing graph, where $\mathcal{E}_s$ is the sensing edge map. Finally, $\mathcal{H} = \{\mathcal{V}, \mathcal{E}_h\}$ is a directed control graph with the set of nodes $\mathcal{V}$ and the control edge map $\mathcal{E}_h$.

Several graphs are needed to capture the interactions of the agents within the network and environment. In some cases, agents can hear but not see each other. The design of the control graph $\mathcal{H}$ involves the assignment of control policies for each agent. The set $\mathcal{E}_h$ is related to the communication $\mathcal{E}_c$ and sensing $\mathcal{E}_s$ graphs. An edge between two nodes in the control graph can be created only if a communication edge or sensing edge exists. Moreover, agents can be modeled as hierarchical hybrid systems [32] that can exchange data that affect their continuous-time motions at discrete-time instants. Figure 14 illustrates how the notation above can describe a dynamic network Σ of vehicles executing a formation control task.

For a dynamic network Σ performing a cooperative task, a motion-coordination algorithm is the control, sensing, and communication law that accomplishes the given task. In the discussion that follows, we assume that all robots are modeled using the model (1), (2), and (4). This model defines X_i, U_i, and f_i for each agent i in the set $\mathcal{A}$. Several motion-coordination algorithms that are implemented on COMET are described.

Formation Control

Many natural systems such as swarms, schools, and flocks exhibit stable formation behaviors [35]. In these systems, individuals follow distant leaders without colliding with neighbors. In some application domains, a group of agents need to move as a *rigid* structure. Also, in practical situations such as cooperative manipulation, a target formation must be established for a given task or environment. In

these cases, reconfiguration of agents in formation is required [24].

Formation control is useful when a network of mobile agents is required to follow a prescribed trajectory $g(t)$ while achieving and maintaining a desired formation shape $\mathcal{F}^d$ [36], [37]. A formation shape $\mathcal{F}$ is described by the relative positions of the agents with respect to a reference coordinate frame. Let the Euclidean distance $\ell_{ij}(t) \in [0, \infty)$ and the bearing $\psi_{ij}(t) \in (-\pi, \pi]$ between agents i and j be defined as

$$\ell_{ij}(t) := \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}, \tag{9}$$

$$\psi_{ij}(t) := \pi + \zeta_{ij}(t) - \theta_i(t), \tag{10}$$

with $\zeta_{ij}(t) = \arctan2(y_i - y_j, x_i - x_j)$, as shown in Figure 15. The following definition is based on [38].

Definition: A *formation* is a network of N vehicles Σ interconnected by means of a collection $S = \{s_{ij}\}$ of node specifications

$$s_{ij}(t) := \begin{bmatrix} \ell_{ij}(t) & \psi_{ij}(t) \end{bmatrix}^T,$$

where $i, j \in \mathcal{I}$, and ℓ_{ij} and ψ_{ij} are the separation distance and bearing angle between vehicles i and j, respectively. An element $s_{ij}(t) \in S$ indicates the relative position vector that follower agent j maintains with respect to its leader agent i. The interconnections between agents are modeled as edges in the directed control graph $\mathcal{H}$ described above and shown in Figure 14.

Note that a desired formation shape $\mathcal{F}^d$ is mapped into a collection $S^d = \{s_{ij}^d\}$ of node specifications, where s_{ij}^d defines a desired separation ℓ_{ij}^d and bearing ψ_{ij}^d for follower agent j with respect to its leader agent i. Both communication-based and vision-based formation control techniques have been implemented on COMET. These algorithms are discussed below.

Vision-based Formation Control

Let us consider a vehicle formation in which a lead vehicle is tele-operated or is required to follow a given trajectory. To implement a formation controller, each robot with the possible exception of the lead robot must have information about the relative location of at least one of its neighbors. This information can either be gathered through communication, if an edge exists in $\mathcal{G}_c$, or sensing, if an edge exists in $\mathcal{G}_s$. Here, we assume that intervehicle communication is disabled and thus no edges exist in the communication graph $\mathcal{G}_c$. However, an edge exists in the sensing graph $\mathcal{G}_s$ from nodes i to j whenever agent i recognizes the NameTag of robot j. COMET uses a decentralized vision-based formation-control strategy that requires knowledge of only leader-follower relative distances and bearing angles. These data are estimated based on measurements from the pan-controlled cameras on board each vehicle. The remainder of this subsection gives an overview of the formation control strategy.

Let the node specification $s_{ij}(t) \in \mathbb{R} \times (-\pi, \pi]$ of agent j be given by (11). Differentiating s_{ij} using (9) and (10), the relative kinematic model becomes

$$\begin{aligned}
\dot{\ell}_{ij} &= v_j \cos\gamma - v_i \cos\psi_{ij}, \\
\dot{\psi}_{ij} &= \frac{v_i \sin\psi_{ij} - v_j \sin\gamma}{\ell_{ij}} - \omega_i, \\
\gamma &= \psi_{ij} + \theta_i - \theta_j,
\end{aligned}$$

where v_i and ω_i are the leader's translational and rotational velocities, respectively. Similarly, the follower's translational and rotational velocities are denoted by v_j and ω_j, respectively. The auxiliary relative angle γ as well as $\ell_{ij}, \psi_{ij}, \theta_i, \theta_j$ are defined in Figure 15. In matrix form, we have

$$\dot{s}_{ij} = \begin{bmatrix} \dot{\ell}_{ij} \\ \dot{\psi}_{ij} \end{bmatrix} = \begin{bmatrix} \cos\gamma & 0 \\ -\frac{\sin\gamma}{\ell_{ij}} & 0 \end{bmatrix} u_j + \begin{bmatrix} -\cos\psi_{ij} & 0 \\ \frac{\sin\psi_{ij}}{\ell_{ij}} & -1 \end{bmatrix} u_i.$$

The objective is to design a control law $u_j(t) = [v_j \quad \omega_j]^T$ that allows agent j to track agent i

with a desired specification $s_{ij}^{d}(t)$, assuming that agent i is stably tracking a desired trajectory $g(t)$ with smooth, bounded inputs $u_i(t) = [v_i \quad \omega_i]^T$. To solve this problem, a high-gain observer is used to estimate the derivatives of the vehicles' relative positions [39].

Communication-Based Formation Control

In communication-based formation control, vehicles follow each other based on their position relative to a common world frame. This behavior requires that each vehicle communicate its position to its neighbors. A robot in the formation is designated as the lead vehicle. The lead vehicle can be driven by a high-level controller or tele-operated through the GUI. Each one of the other robots, on the other hand, is programmed to follow the location communicated by a preselected neighboring robot. This technique requires that an edge be maintained in the communication graph $\mathcal{G}_c$ between two robots in a leader-follower configuration. Sensing is not used, and thus no edges exist in $\mathcal{G}_s$. Communication-based following is useful in urban environments, where the leader agent might not be visible at all times. The success of the technique depends on the accuracy of the position-estimation system. Additionally, the vehicles must share an absolute position reference frame. Figure 16 depicts a group of TXT vehicles changing from a convoy formation to a *V*-like formation and returning to a convoy formation.

Flocking

The mechanisms through which natural systems move together in a decentralized, coordinated manner is a research topic in biology, mathematics, and controls. Flocking is a collective behavior that emerges when a large number of mobile agents with a common goal [22] interact with each other to reach a consensus state in their heading angles and intervehicle separation distances [40]. A simple flocking model is discussed in [41], where several agents travel at the same speed. Each agent aligns its heading based on the average of its heading and the heading of its surrounding neighbors. Despite the absence of

a centralized coordination algorithm, the group aligns itself in the same direction, and flocking behavior emerges. A theoretical explanation for this behavior is presented in [42].

The model described in [41] is implemented on COMET, where each agent is aware of the heading of its neighboring peers. A local PI controller enforces the neighboring rule that aligns the agent's heading according to

$$\theta_i(t+1) = \frac{1}{1+n_i(t)} \left(\theta_i(t) + \sum_{j \in \mathcal{N}_i(t)} \theta_j(t) \right), \tag{11}$$

where θ_i is the heading of agent i, and $n_i(t)$ is the number of neighbors that can be heard or seen by agent i at time t. Also, all of the vehicles are commanded to maintain the same translational speed $v_{\mathcal{I}}^c(t)$. Although obstacle avoidance is active, the implementation is tested in an obstacle-free environment. Figure 17 shows a five-agent team in which one of the robots follows a set of waypoints to provide direction for the flock. The remaining robots use the alignment rule (11) to successfully navigate together through the desired waypoints.

Perimeter Detection and Tracking

We now describe a decentralized coordination algorithm that allows a mobile sensor network to find and track a dynamic perimeter. A perimeter can be a chemical substance spill, building, or landmark. We assume that each agent is equipped with an appropriate sensor to detect the perimeter. As before, we consider a network Σ of N mobile agents, each modeled with the unicycle model with control input $u_i^c = [v_i^c \quad \omega_i^c]^T \in U$, where U is a bounded, convex set.

We define a convex polygonal searching area $\mathcal{S} \subset \mathbb{R}^2$ with boundary $\partial\mathcal{S}$ containing a target region $\Omega \subset \mathcal{S}$ with perimeter $\partial\Omega$. The robots are initially placed in the environment at random locations $(x_i, y_i) \in \mathcal{S} \setminus \Omega$, as shown in Figure 18(a). Also, agents can communicate with other members of the team at discrete time instants. Finally, we assume that each agent has a limited field of view and limited

communication range. Now we state the perimeter-detection and tracking problem as follows.

Perimeter Detection and Tracking Problem: Given a dynamic network Σ of N mobile sensors randomly distributed in a planar polygonal environment $\mathcal{S}$, determine a motion-coordination algorithm such that the team detects and tracks a perimeter $\partial\Omega$ defined by a target region Ω while avoiding intervehicle collisions.

To solve this problem, we implement a motion-coordination algorithm composed of a random-coverage controller to start the search for the perimeter, a PFC (discussed above) to attract agents to a known perimeter boundary point, and a tracking controller to track the perimeter. The agents coordinate their efforts to locate the perimeter. Once an agent finds the perimeter containing the target region, the agent broadcasts its position to all robots within range. As each robot receives the message, it starts the PFC to reach the perimeter location and retransmits the received message, creating a communication relay. Additionally, OA is active in the background throughout the execution of the algorithm to prevent collisions. The goal of this algorithm is to use a multirobotic system to estimate a dynamic perimeter as it evolves. Once an estimate of the perimeter is known, cooperation can be used to uniformly surround the target region [43]. Figure 18 shows the algorithm in operation.

A Target Assessment Case Study

This section demonstrates how COMET might be used in the real world for surveillance or target assessment. The experiment uses basic behaviors, motion-coordination algorithms, and the GUI described previously. Figure 19 shows the site where the experiment is conducted.

The three phases that comprise the mission include securing the path, driving to the target, and encircling the target. These steps illustrate the modes of operation and levels of autonomy discussed in the Robot Control Software section above.

Securing the Path

For the convoy to travel safely, the path must be secured. When the mission begins, two *scout* agents position themselves at tactical locations to survey the path. Figure 19 shows those locations, which are labeled as the surveillance points. Each scout is commanded to its surveillance point using a series of waypoints. Once its surveillance point is reached, the scout agent pans its camera to provide a panoramic view to users, allowing them to verify the safety of the path.

After the first scout reaches the first surveillance location, the second scout drives farther into the preselected convoy path. Like the first scout, its function is to monitor the area and provide video surveillance feedback.

Driving to the Target

Once the path is known to be safe, the convoy is commanded to go to the target location. The convoy is composed of one leader agent and three followers. The first agent in the convoy is provided with a series of waypoints to reach the target. The remaining agents follow the leader, executing a communication-based leader-follower operation.

Encircling the Target

Once the convoy has reached the vicinity of the target, each vehicle is commanded to a new location. This operation is based on a series of preselected waypoints that guide the robot to a particular location near the target. The objective of this rearrangement is to provide the user with a diverse view of the target. The user can now, by means of remote video, assess the target condition.

Conclusions

A scalable multivehicle platform has been developed to demonstrate cooperative control systems and sensor networks. Major features of this platform include high flexibility to varying sensor configurations, with a pool of hot-swappable sensors that can be added or removed on-the-fly, and a modular framework for integrating sensing and control capabilities. The platform is compatible with Player and Gazebo, which makes it easily accessible and offers the benefit of a simulation environment with full sensor support. The software library gives developers a basic set of building blocks for more complex algorithms.

Currently, we are adding motion-coordination algorithms to the library of team controllers. These algorithms include deployment, rendezvous [44], perimeter estimation and pattern formation [45], and dynamic target tracking [46]. We are also exploring alternative methods, such as machine learning [47] and optimal formation shape changes [48], to improve the performance of existing motion-coordination algorithms. Additionally, new sensing devices and drivers are being added to the suite of onboard, hot-swappable embedded sensors. These devices include NorthStar localization systems from Evolution Robotics, environmental wireless sensors from Moteiv, and fisheye lens cameras.

COMET provides a comprehensive research and educational tool, allowing students to learn hardware and software design for embedded computer systems and become familiar with cutting-edge computer-controlled design and manufacturing technology. Additionally, COMET provides researchers in cooperative and distributed control with an interdisciplinary environment to integrate embedded sensing, hybrid control, and communication networks.

Acknowledgments

The authors would like to thank the anonymous reviewers for providing useful comments and

suggestions that improved the presentation of this paper. We also thank Justin Clark for his work in perimeter detection, Rahul Buddhisagar for his effort assisting in the development of the graphical user interface, Chris Flesher for his work in computer vision, and Colby Toland and Kyle Hutchins for their efforts in tuning the PID controllers. This work is supported in part by NSF grants #0311460 and CAREER #0348637 and by the U.S. Army Research Office under grant DAAD19-03-1-0142 (through the University of Oklahoma).

References

[1] D. Hristu-Varsakelis and W. S. Levine, Eds., *Handbook of Networked and Embedded Control Systems*, ser. Control Engineering. Boston: Birkhäuser, 2005.

[2] Y. Cao, S. Fukunga, and A. Kahng, "Cooperative mobile robotics: Antecedents and directions," *Autonomous Robots*, vol. 4, pp. 1–23, 1997.

[3] (2007, January) Marhes multimedia. [Online]. Available: http://marhes.okstate.edu/multimedia.htm

[4] Z. Jin, S. Waydo, E. B. Wildanger, M. Lammers, H. Scholze, P. Foley, D. Held, and R. M. Murray, "MVWT-II: The second generation caltech multi-vehicle wireless testbed," in *Proc. American Control Conf.*, vol. 6, Boston, MA, June 2004, pp. 5321–5326.

[5] V. Vladimerou, A. Stubbs, J. Rubel, A. Fulford, and G. Dullerud, "Multivehicle systems control over networks," in *IEEE Control Systems Magazine*, vol. 26, no. 3, June 2006, pp. 56–69.

[6] E. King, Y. Kuwata, M. Alighanbari, L. Bertuccelli, and J. How, "Coordination and control experiments on a multi-vehicle testbed," in *Proc. American Control Conf.*, vol. 6, Boston, MA, June 2004, pp. 5315–5320.

[7] L. Chaimowicz, B. Grocholsky, J. F. Keller, V. Kumar, and C. J. Taylor, "Experiments in multirobot air-ground coordination," in *Proc. IEEE Int. Conf. Robot. Automat.*, vol. 4, New Orleans, LA, April 2004, pp. 4053–4058.

[8] B. Grocholsky, J. Keller, V. Kumar, and G. Pappas, "Cooperative air and ground surveillance," *IEEE Robotics & Automation Magazine*, vol. 13, no. 3, pp. 16–26, September 2006.

[9] R. D'Andrea and M. Babish, "The RoboFlag testbed," in *Proc. American Control Conf.*, vol. 1, Boston, MA, June 2004, pp. 656–660.

[10] T. W. McLain and R. W. Beard, "Unmanned air vehicle testbed for cooperative control experiments," in *Proc. American Control Conf.*, vol. 6, Boston, MA, June 2004, pp. 5327–5331.

[11] T. Balch and R. Arkin, "Behavior-based formation control for multirobot teams," *IEEE Trans. on*

Robotics and Automation, vol. 14, no. 46, pp. 926–939, December 1998.

[12] M. Egerstedt, *Control of Autonomous Mobile Robots*, ser. Control Engineering Series, D. Hristu-Varsakelis and W. Levine, Eds. Boston, MA: Birkhauser, 2005.

[13] P. Ögren, E. Fiorelli, and N. E. Leonard, "Cooperative control of mobile sensor networks: Adaptive gradient climbing in a distributed environment," *IEEE Trans. on Automatic Control*, vol. 49, no. 8, pp. 1292–1302, August 2004.

[14] K. Kyriakopoulos, P. Kakambouras, and N. Krikelis, "Navigation of nonholonomic vehicles in complex environments with potential fields and tracking," in *Proc. IEEE Int. Conf. Robot. Automat.*, Minneapolis, Minnesota, USA, April 1996, pp. 2968–2973.

[15] J. Sanchez and K. Walling. (2004, July) MARHES TXT-1 CAN bus control unit. [Online]. Available: http://marhes.okstate.edu/CAN_control_unit.htm

[16] B. Gerkey, R. T. Vaughn, and A. Howard, "The Player/Stage project: Tools for multi-robot and distributed sensor systems," in *Proc. of the 11th Int. Conf. on Advanced Robotics*, Coimbra, Portugal, June 2003, pp. 317–323.

[17] K. Etschberger, *Controller Area Network: Basics, Protocols, Chips and Applications*. Weingarten, Germany: IXXAT Press, 2001.

[18] A. De Luca, G. Oriolo, and C. Samson, "Feedback control of a nonholonomic car-like robot," in *Robot Motion Planning and Control*, J.-P. Laumond, Ed. London: Springer-Verlag, 1998, pp. 171–253.

[19] L. Ojeda, D. Cruz, G. Reina, and J. Borenstein, "Current-based slippage detection and odometry correction for mobile robots and planetary rovers," *IEEE Trans. on Robotics*, vol. 22, no. 2, pp. 366–378, April 2006.

[20] O. Orqueda, "Vision-based control of multi-agent systems," Ph.D. Thesis, Oklahoma State University, 202 Engineering South, Stillwater, OK 74078, Dec. 2006. [Online]. Available: http://marhes.okstate.edu/dissertations/orqueda_06.pdf

[21] L. Mirkin and Z. J. Palmor, "Control issues in systems with loop delays," in *Handbook of Networked and Embedded Control Systems*, D. Hristu-Varsakelis and W. S. Levine, Eds. Birkhäuser, 2005, pp. 627–649.

[22] R. Olfati-Saber, "Flocking for multi-agent dynamic systems: Algorithms and theory," *IEEE Trans. on Automatic Control*, vol. 51, no. 3, pp. 401–420, March 2006.

[23] R. Fierro, A. Das, V. Kumar, and J. P. Ostrowski, "Hybrid control of formations of robots," in *Proc. IEEE Int. Conf. Robot. Automat.*, Seoul, Korea, May 2001, pp. 157–162.

[24] R. Fierro, L. Chaimowicz, and V. Kumar, "Multi-robot cooperation," in *Autonomous Mobile Robots: Sensing, Control, Decision Making and Applications*, ser. Control Engineering, S. S. Ge and F. L. Lewis, Eds. CRC Press - Taylor & Francis Group, May 2006, ch. 11, pp. 417–459.

[25] D. Terzopoulos and R. Szeliski, *Active Vision*. Cambridge, MA: MIT Press, 1992, ch. Tracking with Kalman Snakes, pp. 3–20.

[26] P. Renaud, E. Cervera, and P. Martinet, "Towards a reliable vision-based mobile robot formation control," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, vol. 4, 2004, pp. 3176–3181.

[27] N. Otsu, "A threshold selection method from gray-level histograms," *IEEE Trans. on Syst., Man, and Cyber.*, vol. 9, no. 1, pp. 62–66, 1979.

[28] D. Lowe, "Fitting parameterized three-dimensional models to images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, no. 5, pp. 441–450, 1991.

[29] D. Oberkampf, D. DeMenthon, and L. Davis, "Iterative pose estimation using coplanar feature points," *Computer Vision and Image Understanding*, vol. 63, no. 3, pp. 495–511, May 1996.

[30] J. C. Latombe, *Robot Motion Planning*. Boston, Massachusetts USA: Kluwer Academic Publishers, 1991.

[31] J. Borenstein and U. Raschke, "Real-time obstacle avoidance for non-point mobile robots," in *SME Transactions on Robotics Research*, vol. 2, September 1992, pp. 2.1–2.10.

[32] R. Alur, T. Dang, J. Esposito, R. Fierro, Y. Hur, F. Ivancic, V. Kumar, I. Lee, P. Mishra, G. Pappas,

and O. Sokolsky, "Hierarchical hybrid modeling of embedded systems," in *Embedded Software*, ser. LNCS 2211, T. Henzinger and C. Kirsch, Eds. Springer, 2001, pp. 14–31.

[33] S. Martínez, F. Bullo, J. Cortés, and E. Frazzoli, "On synchronous robotic networks. part i: Models, tasks and complexity notions," in *Proc. IEEE Conf. on Decision and Control, and the European Control Conf.*, Seville, Spain, December 12-15 2005, pp. 2847–2852.

[34] N. A. Lynch, *Distributed Algorithms*. San Mateo, CA: Morgan Kaufmann Publishers, 1997.

[35] J. Toner and Y. Tu, "Flocks, herds and schools: A quantitative theory of flocking," *Physical Review E*, vol. 58, no. 4, pp. 4828–4858, 1998.

[36] G. Lafferriere, A. Williams, J. Caughman, and J. Veerman, "Decentralized control of vehicle formations," *Systems & Control Letters*, vol. 54, no. 9, pp. 899–910, September 2005.

[37] A. K. Das, R. Fierro, V. Kumar, J. P. Ostrowski, J. Spletzer, and C. J. Taylor, "A vision-based formation control framework," *IEEE Trans. on Robotics and Automation*, vol. 18, no. 5, pp. 813–825, October 2002.

[38] H. Tanner, V. Kumar, and G. Pappas, "Leader-to-formation stability," *IEEE Trans. on Robotics and Automation*, vol. 20, no. 3, pp. 443–455, June 2004.

[39] O. A. Orqueda and R. Fierro, "Robust vision-based nonlinear formation control," in *Proc. American Control Conf.*, Minneapolis, MN, June 14-16 2006, pp. 1422–1427.

[40] N. Moshtagh, A. Jadbabaie, and K. Daniilidis, "Distributed geodesic control laws for flocking of nonholonomic agents," in *Proc. IEEE Conf. on Decision and Control and European Control Conference (CDC-ECC'05)*, Seville, Spain, December 2005, pp. 2835–2840.

[41] T. Vicsek, A. Czirok, E. B. Jacob, I. Cohen, and O. Schochet, "Novel type of phase transitions in a system of self-driven particles," in *Physical Review Letters*, vol. 75, 1995, pp. 1226–1229.

[42] J. L. A. Jadbabaie and A. S. Morse, "Coordination of groups of mobile autonomous agents using nearest neighbor rules," in *IEEE Transactions on Automatic Control*, 2003, pp. 988–1001.

[43] J. Clark and R. Fierro, "Cooperative hybrid control of robotic sensors for perimeter detection and

tracking," in *Proc. American Control Conf.*, Portland, OR, June 8-10 2005, pp. 3500 – 3505.

[44] A. Ganguli, S. Susca, S. Martínez, F. Bullo, and J. Cortés, "On collective motion in sensor networks: sample problems and distributed algorithms," in *Proc. IEEE Conf. on Decision and Control*, Seville, Spain, December 2005, pp. 4239–4244.

[45] M. A. Hsieh and V. Kumar, "Pattern generation with multiple robots," in *Proc. IEEE Int. Conf. Robot. Automat.*, Orlando FL, May 15-19 2006, pp. 2442–2447.

[46] T. H. Chung, J. W. Burdick, and R. M. Murray, "A decentralized motion coordination strategy for dynamic target tracking," in *Proc. IEEE Int. Conf. Robot. Automat.*, Orlando, Florida, May 2006, pp. 2416–2422.

[47] M. Lomas, P. Vernaza, D. Lee, and V. Kumar, "Hill-climbing for potential field based cooperative navigation," in *Proc. IEEE Int. Conf. Robot. Automat.*, Orlando, Florida, May 2006, pp. 4318–4320.

[48] J. Spletzer and R. Fierro, "Optimal position strategies for shape changes in robot teams," in *Proc. IEEE Int. Conf. Robot. Automat.*, Barcelona, Spain, April 18-22 2005, pp. 754–759.

Sidebar: COMET Components and vendors

Item: Chassis

Model: Tamiya TXT-1

Vendor: Tower Hobbies (http://www.towerhobbies.com)

Item: PC-104 (embedded computer)

Model: ADL855PC

Vendor: Advanced Digital-Logic (http://www.adlogic-pc104.com)

Item: Sensors (IR, LRF, Ultrasonic ranger)

Model: Sharp GP2Y0A02YK, Hokuyo URG-04LX, Devantech SRF08

Vendor: Acroname (http://www.acroname.com)

Item: IEEE-1394 stereo camera

Model: Bumblebee

Vendor: Point Grey Inc. (www.ptgrey.com)

Item: PC-CAN interface

Model: USBcan, LAPcan

Vendor: KVASER (http://www.kvaser.com)

Item: GPS

Model: GPS18 5Hz

Vendor: Garmin (http://www.garmin.com)

Item: IMU

Model: 3DM-GX1

Vendor: MicroStrain Inc. (http://www.microstrain.com)

Daniel Cruz received a bachelor degree in electronics engineering from the Pontificia Javeriana University, Colombia in 2001. He worked at the University of Michigan as a visitor research investigator for 2 years and received a master's degree in electrical and computer engineering at Oklahoma State University, Stillwater in 2006. His areas of interest include positioning systems, mobile robot navigation, and hardware integration.

James McClintock received a B.S. degree in electrical engineering from Oklahoma State University, Stillwater, Oklahoma in 2006. His research interests include cooperative control of robotic systems and development of advanced robot-human interfaces. In 2004, he was awarded a NASA Oklahoma Space Grant Consortium research fellowship to develop a robot-human interface for the Evolution Robotics Scorpion Robot. He is currently pursing a master's degree in electrical engineering at OSU.

Brent Perteet received a B.S. degree in electrical engineering from Oklahoma State University in 2005, where he is currently pursuing the M.S. degree in electrical engineering. His research interests include cooperative control of robotic agents, strategies for pursuit-evasion and target tracking, embedded systems, and software engineering.

Omar A.A. Orqueda received a B.S. degree in electronics engineering from the Universidad Nacional del Sur, Bahía Blanca, Argentina. He received the Ph.D. in electrical engineering from the School of Electrical and Computer Engineering, Oklahoma State University, Stillwater, OK. His research interests include distributed coordination of autonomous systems, artificial vision, nonlinear control theory, and motion planning.

Yuan Cao received a B. Eng. and M.S. degrees from Zhejiang University, Hangzhou, China. He is currently pursuing a Ph.D. at the School of Electrical and Computer Engineering, Oklahoma State University, Stillwater, OK. His research interests include cooperative control of unmanned vehicles,

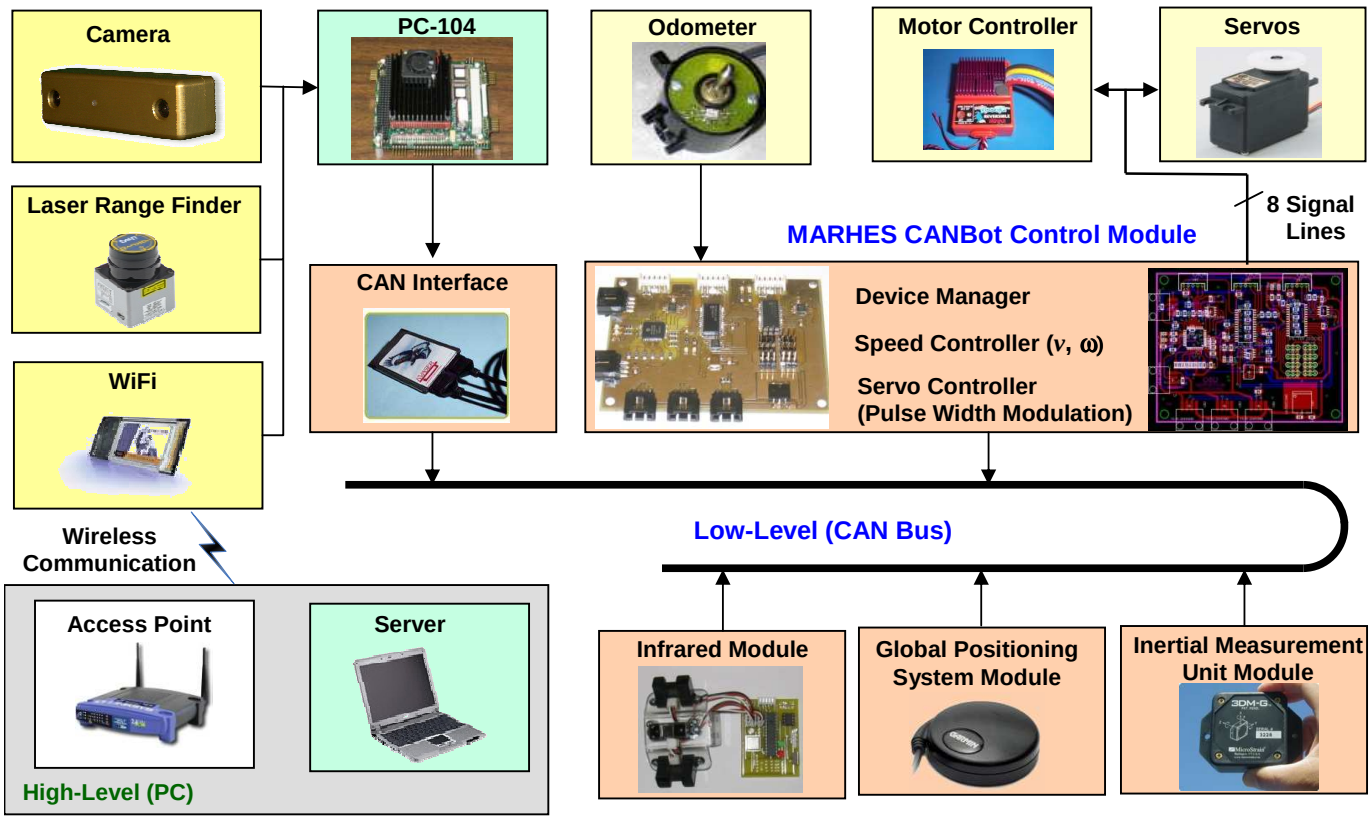
estimation theory and hybrid systems.

Rafael Fierro (rfierro@okstate.edu) received a M.Sc. degree in control engineering from the University of Bradford, England, and a Ph.D. degree in electrical engineering from the University of Texas at Arlington in 1990 and 1997, respectively. From 1999 to 2001, he held a postdoctoral research appointment with the GRASP Lab, University of Pennsylvania. He is currently an assistant professor in the School of Electrical and Computer Engineering at Oklahoma State University. His research interests include hierarchical hybrid and embedded systems, optimization-based cooperative control, and robotics. He is the recipient of a Fulbright Scholarship and a 2004 National Science Foundation CAREER Award. He can be contacted at the School of Electrical and Computer Engineering, Oklahoma State University, 202 Engineering South, Stillwater, OK 74078-5032, USA. Tel. (405) 744-1328. For contact information, visit the MARHES website at http://marhes.okstate.edu.

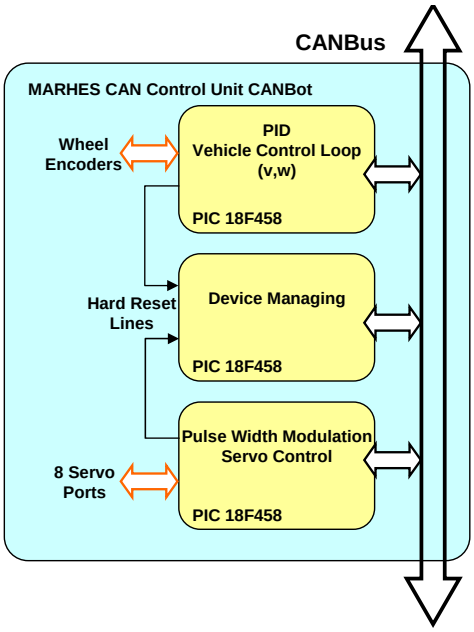
Dimensions	Length	498 mm
	Width	375 mm
	Height	240 mm
Weight	Loaded	8.2 kg
	Unloaded	5.2 kg
Maximum Speed		1.60 m/s
Carry Capacity		3.7 kg
Minimum Turning Radius		1359 mm

TABLE I: Platform mechanical specifications. Each vehicle can carry sensors and a payload. Agents are small enough to be used indoors but robust enough to perform outdoors.

Figure 1: Multivehicle platform at Oklahoma State University. The platform consists of ten wirelessly networked robots based on the TXT-1 monster truck from Tamiya, Inc. Each robot is provided with a control board, odometry, various hot-swappable sensors, and an onboard embedded computer.

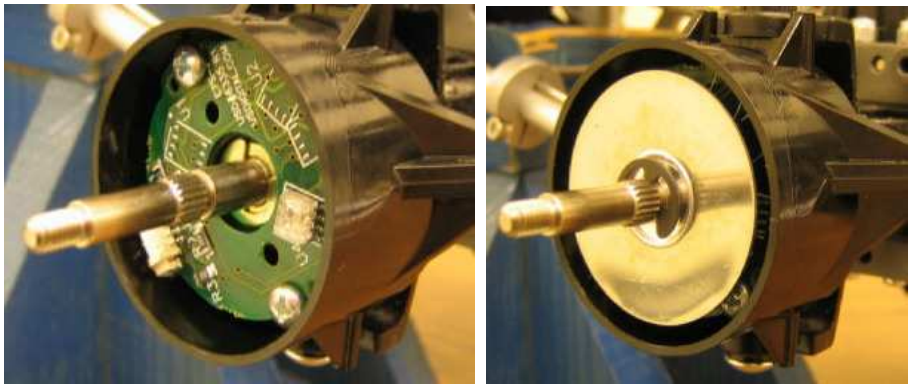


(a)



(b)

Figure 2: (a) In-vehicle controller area network (CAN). (b) CANBot unit block diagram. Vehicle sensors are networked using the CANBus. The CANBot control board hosts the CANBus and provides an interface between sensors and the robot's onboard computer. The camera and laser range finder are connected directly to the computer because of their bandwidth requirements.



(a) (b)

Figure 3: (a) E7MS encoder placement in platform wheel. (b) Code wheel. Each of the vehicle's front wheels is equipped with a miniature encoder from USDigital. By measuring wheel rotation, these sensors are used to determine the translational and rotational speeds of the vehicle.

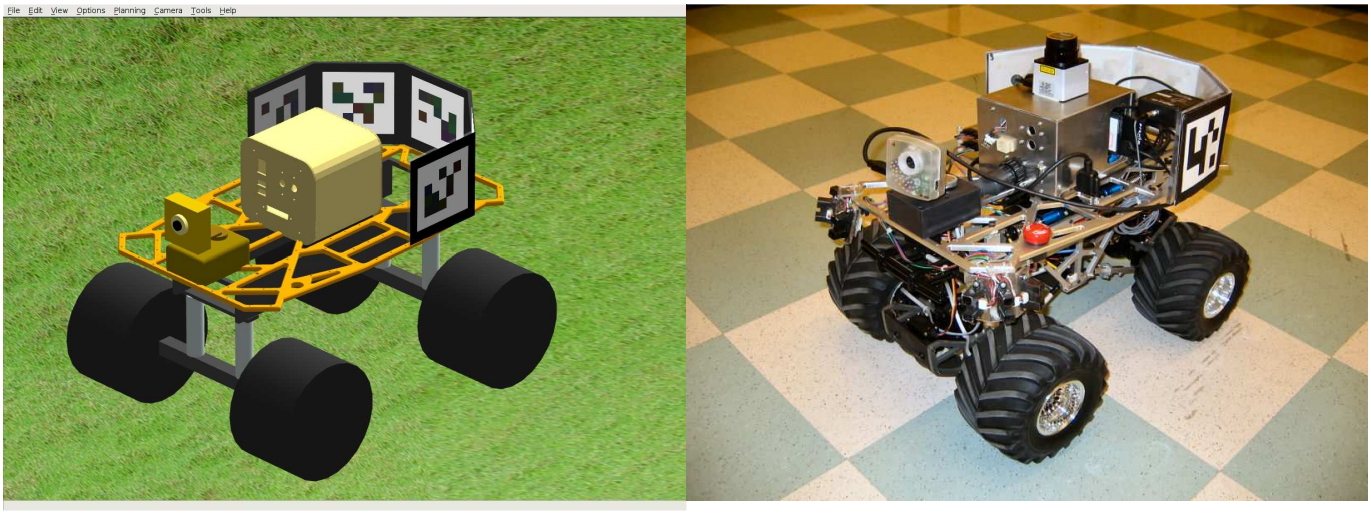
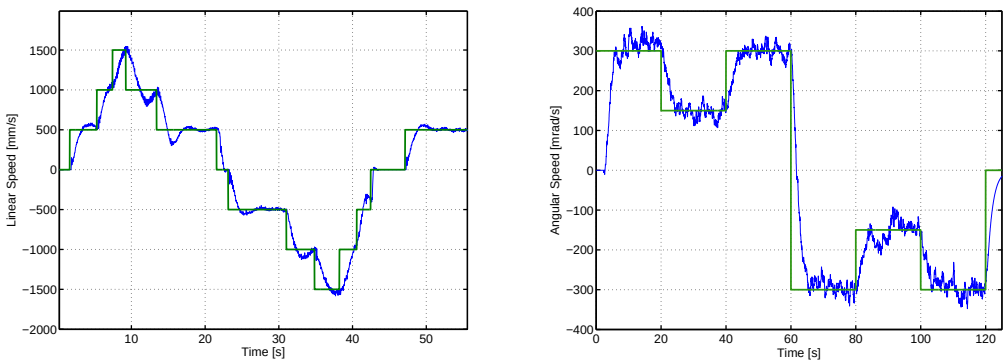


Figure 4: 3D simulation model and assembled vehicle. Sensors are mounted on a plate designed using SolidWorks 2004 and milled using a Haas computer-controlled milling machine. Additional modifications include the installation of an electric speed controller (Novak Super Rooster), foam inserts for the wheels to increase traction, stabilizer bar clamps, and center skid plates to protect the gearbox.



(a) (b)

Figure 5: (a) Translational speed profile tracking. (b) Rotational speed profile tracking. The gains (K_p, K_i, K_d) are adjusted to reach the commanded velocity vector $u_c(t)$ in minimal time with the smallest possible overshoot. The translational speed gains are $(0.8$ s/m$, 1.0$ s/m$, 0.02$ s/m$)$, while the rotational speed gains are $(0.4$ s$, 0.01$ s$, 0.1$ s$)$.

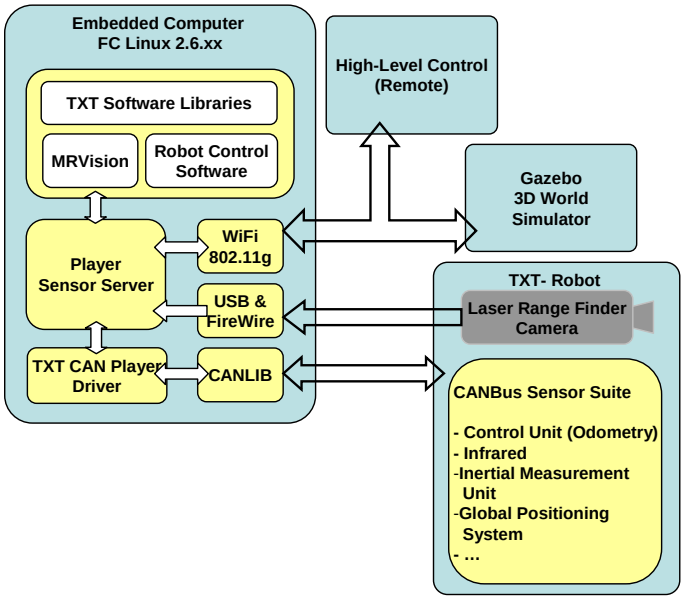
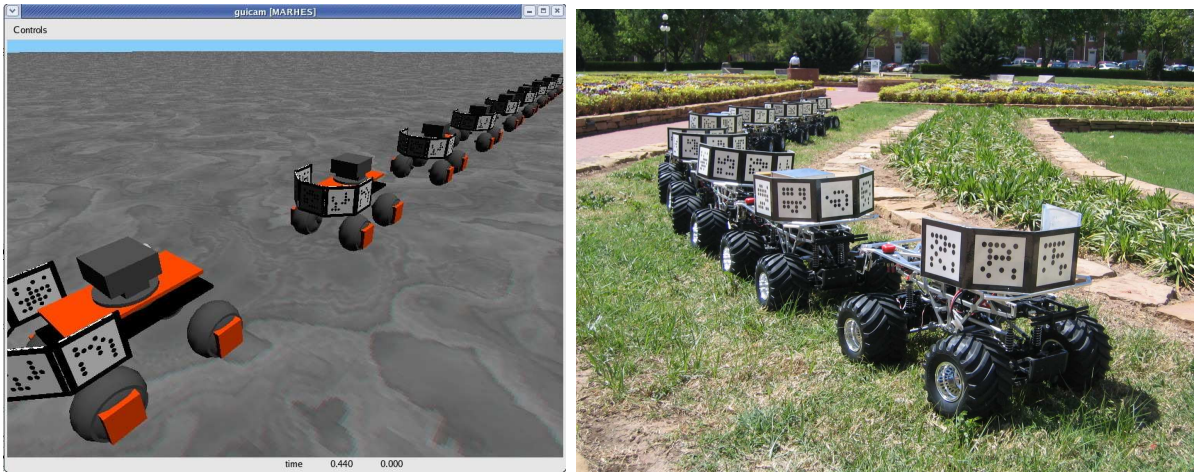


Figure 6: Vehicle software architecture. This block diagram shows how the onboard sensor network interacts with the local embedded computer using the controller area network (CAN). Player runs on the local computer and allows local or remote applications to access the system. A compatible simulation environment is also available.



(a) (b)

Figure 7: (a) A Gazebo simulation of the platform. (b) Real vehicles. This figure shows the similarities between the simulated platform in Gazebo and the physical robots. In fact, software written for the simulated vehicles can run on the robots without modification.

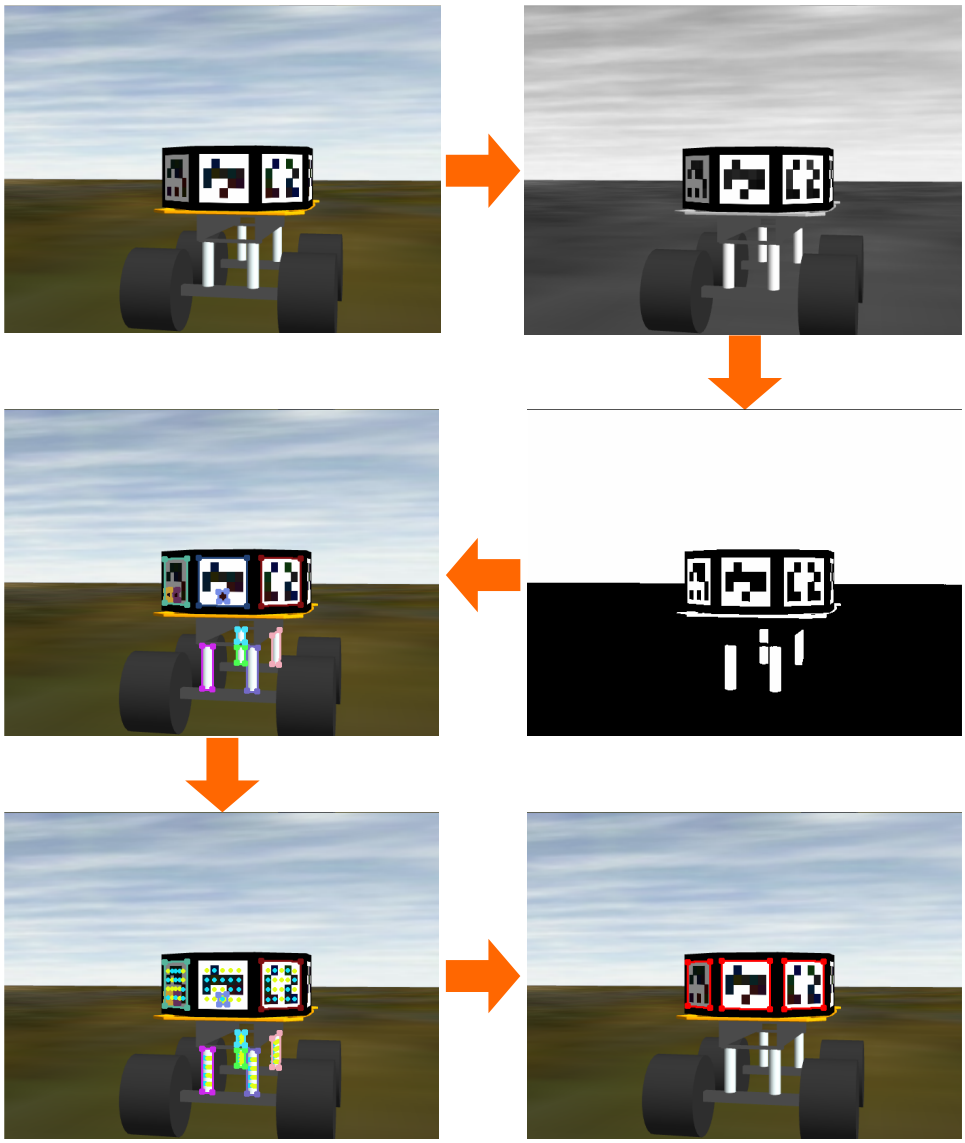


Figure 8: Mobile robot vision processing sequence. The mobile robot vision library uses camera images to recognize NameTags and compute relative position and orientation. This figure shows the sequence of main steps involved in extracting NameTag information from a frame. First, a camera image is captured and converted to grayscale. Next, a thresholding algorithm is applied and contours are located. Each contour is then processed to determine whether it represents a valid face. This library can run on any computer with a camera. The PC-104 executes the algorithm at about 10 frames per second.

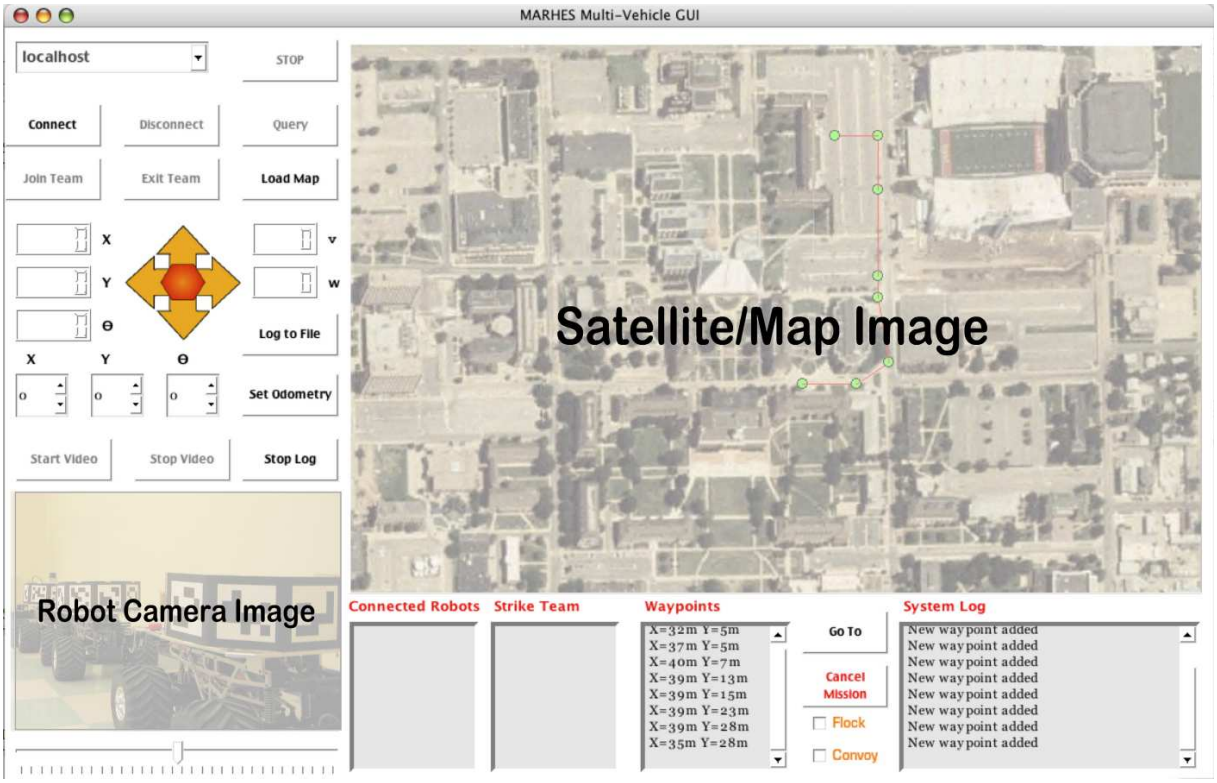


Figure 9: The multivehicle graphical user interface. This program, which runs on a base station, allows monitoring and control of a group of agents. Vehicles can be maneuvered as a group using built-in swarming behaviors such as flocking, or they can be controlled individually.

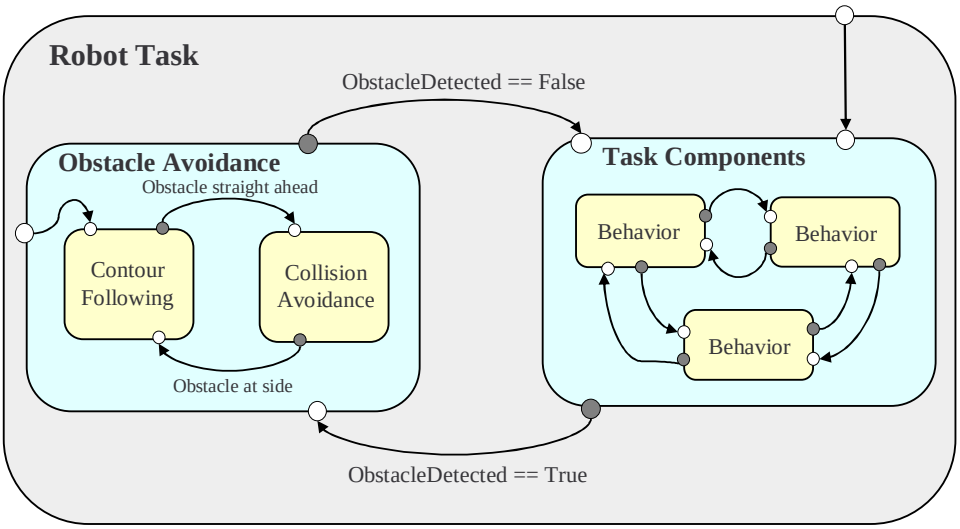
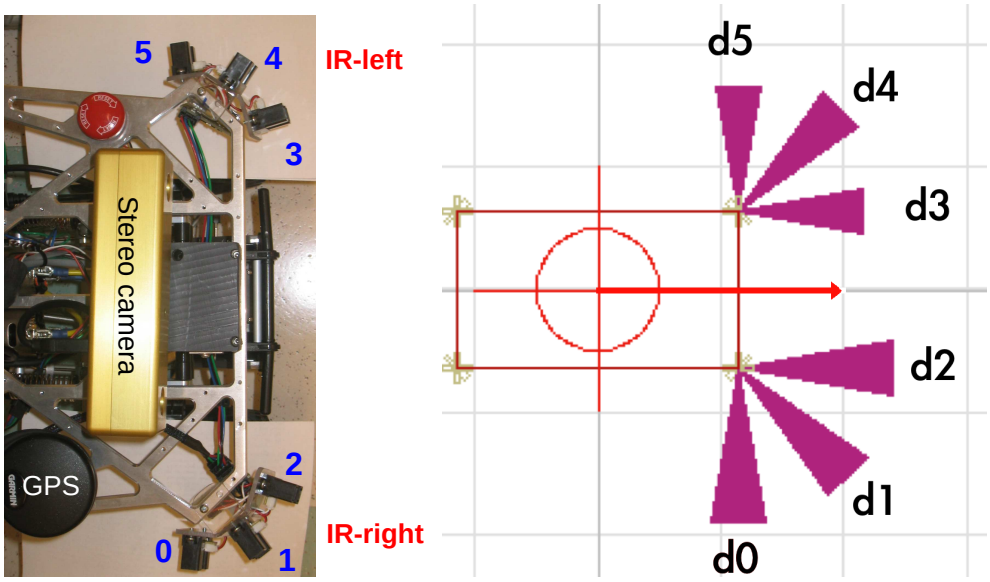


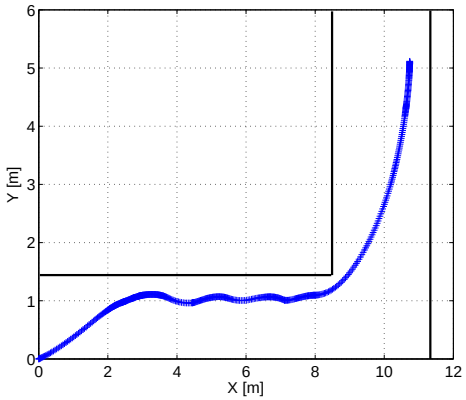
Figure 10: Obstacle avoidance behavior in a hybrid system. In this system, obstacle avoidance is used when an obstacle is detected nearer to the robot than a threshold value. Otherwise, the behaviors that comprise the robot's current task are run.



(a) (b)

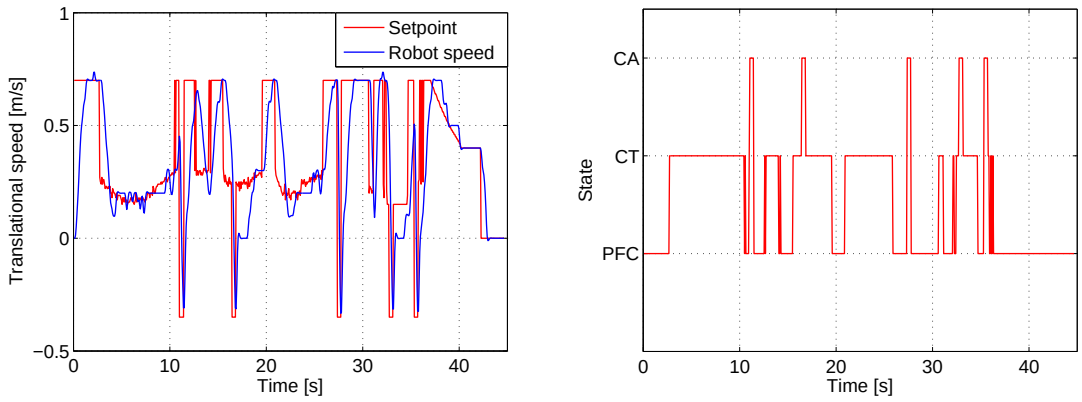
Figure 11: (a) Top-front view of vehicle. (b) Infrared sensor coverage diagram. Infrared sensors are used for obstacle avoidance and contour following. Two infrared arrays are mounted at the front of each robot. Each array provides distance-to-obstacle information straight ahead, at a 45-deg angle, and at a 90-deg angle.

(a)



(b)

Figure 12: (a) Starting position for a goal-seeking experiment. (b) Agent trajectory to goal. The designated goal is located at (11 m, 5.5 m) around the corner in the hallway. As soon as the goal-seeking behavior starts, the potential field controller begins aligning the vehicle toward the goal.



(a) (b)

Figure 13: (a) Goal-seeking experiment vehicle speed plot. (b) State plot. The goal-seeking behavior uses a hybrid controller that switches between potential field control (PFC), contour tracking (CT), and collision avoidance (CA) states as obstacles are encountered and negotiated. During the goal-seeking experiment, the vehicle's speed fluctuates as the controller switches between states while the robot travels toward its goal.

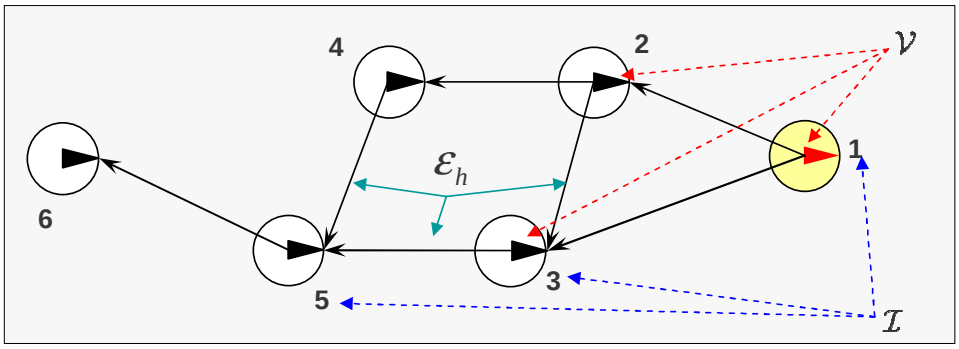
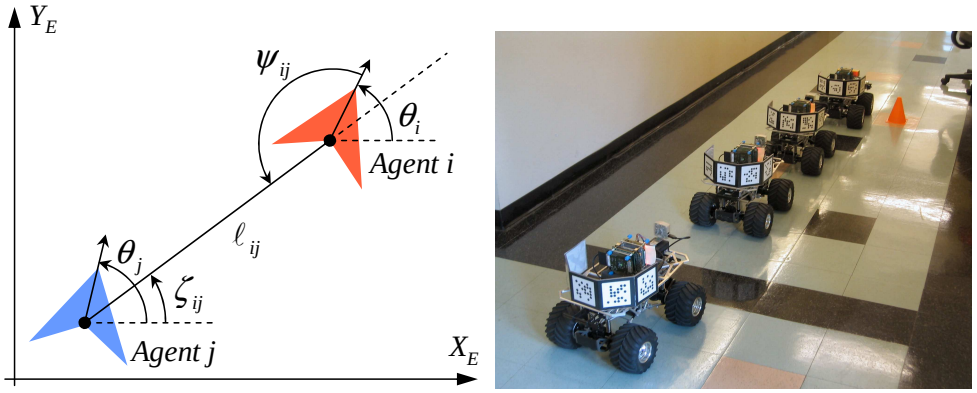
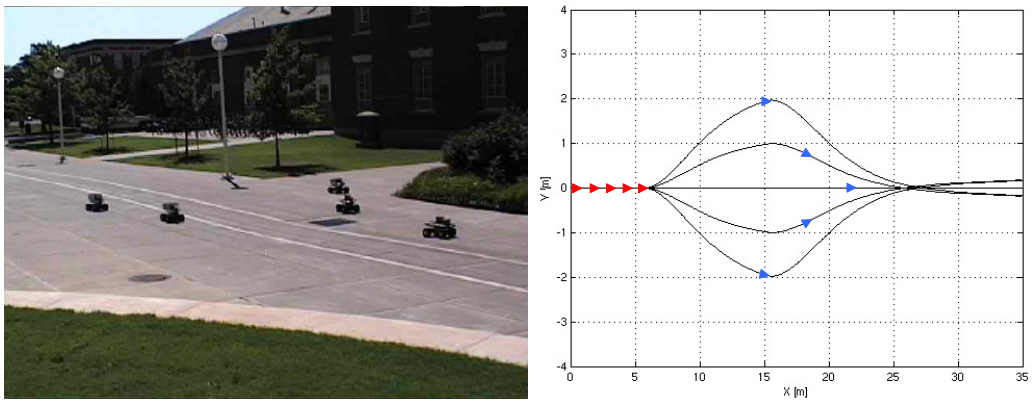


Figure 14: A dynamic network Σ for formation control. Each vehicle, which is assigned a unique identifier in the set $\mathcal{I}$, serves as a node $\mathcal{V}$ in the communication $\mathcal{G}_c$, sensing $\mathcal{G}_s$, and control $\mathcal{H}$ graphs. For simplicity, only the directed control graph $\mathcal{H} = \{\mathcal{V}, \mathcal{E}_h\}$ is shown. To complete the definition of Σ, the set of control systems $\mathcal{A}$ must be specified.



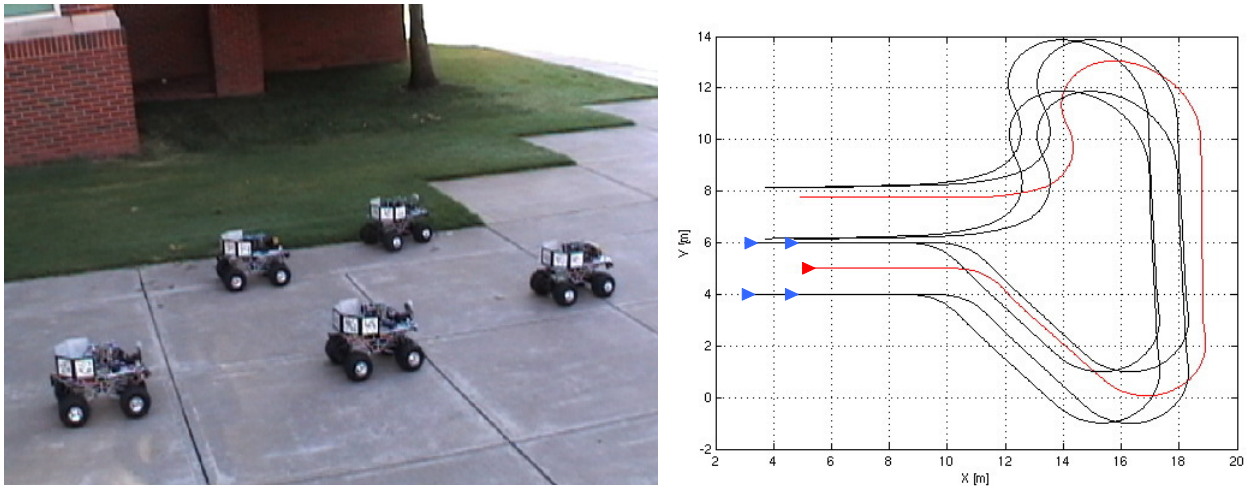
(a) (b)

Figure 15: (a) Formation geometry diagram. (b) A leader-follower experiment. Each agent calculates the position of the vehicle directly in front of it and follows that vehicle in formation. The controller specification for robot j indicates the separation distance ℓ_{ij} and bearing angle ψ_{ij} to be maintained with respect to its leader vehicle i. The lead agent is tele-operated.



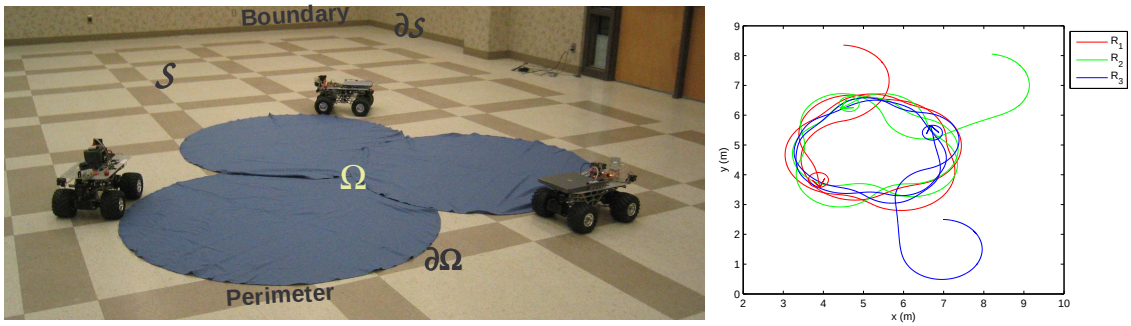
(a) (b)

Figure 16: (a) Communication-based formation control experiment. (b) Vehicle paths. In this experiment, the vehicles change formations while traveling to a goal. The red triangles in (b) represent the vehicles' starting position in a convoy formation. Later they assume a V-like formation as shown in (a) and by the blue triangles in (b). Finally, the robots return to a convoy formation. The maneuver is based on odometry and intervehicle communications.



(a) (b)

Figure 17: (a) Flocking experiment. (b) Agent paths. This figure shows a flocking experiment with five robots. The triangles in (b) represent the robots' starting positions. In pure flocking, each agent runs the same controller and calculates its speed based on the speeds of neighboring agents. During this modified flocking experiment, one of the agents (red path) follows a set of waypoints to provide direction for the flock. The remaining robots use the flocking controller presented in this article, and the team successfully travels together through the desired waypoints.



(a) (b)

Figure 18: (a) Perimeter-detection experiment. (b) Agent paths. The random-coverage controller commands the agents in a spiral search path looking for the perimeter. A blobfinder, part of the vision library, uses the onboard camera to detect and track the perimeter. Once an agent finds the perimeter, the blobfinder-based tracking controller tracks it.

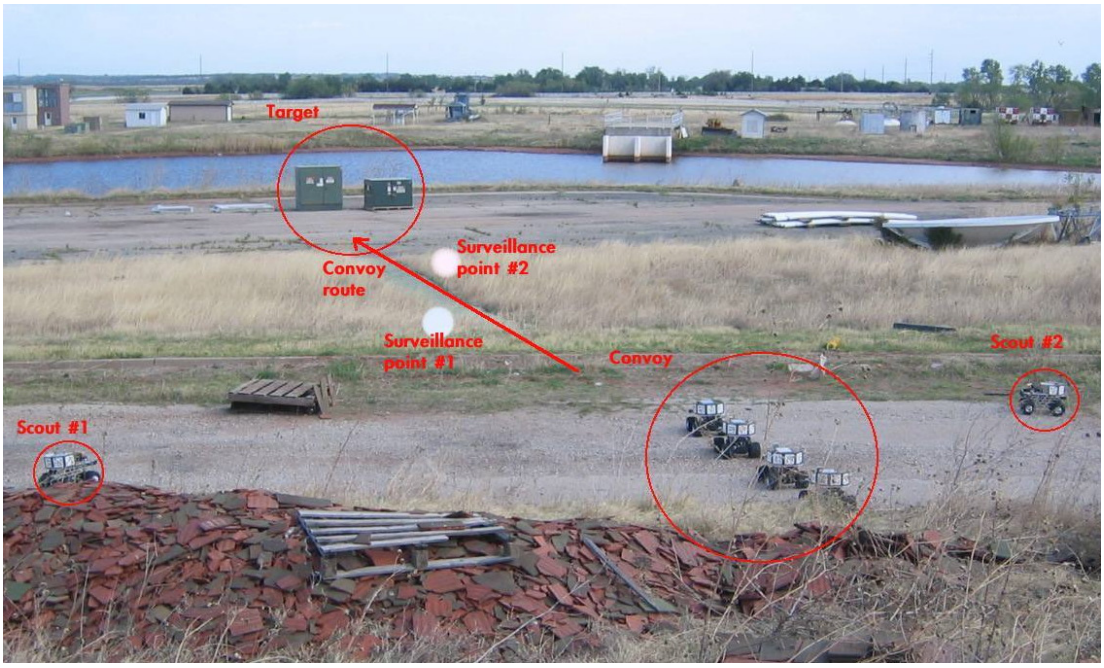
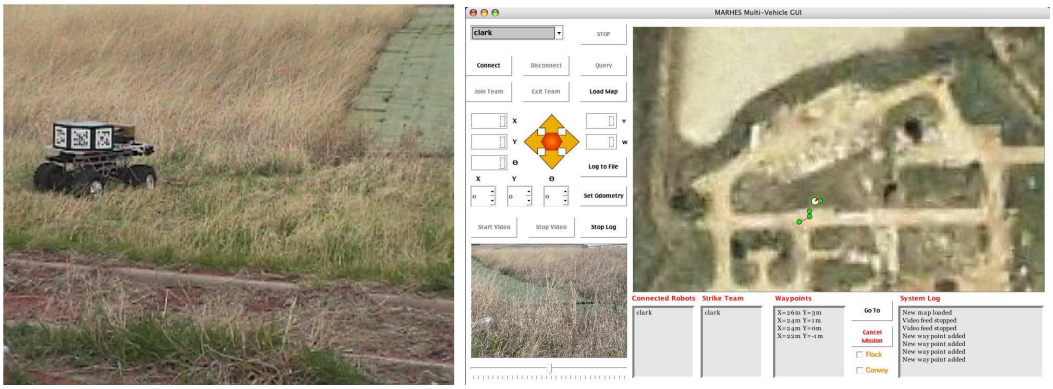
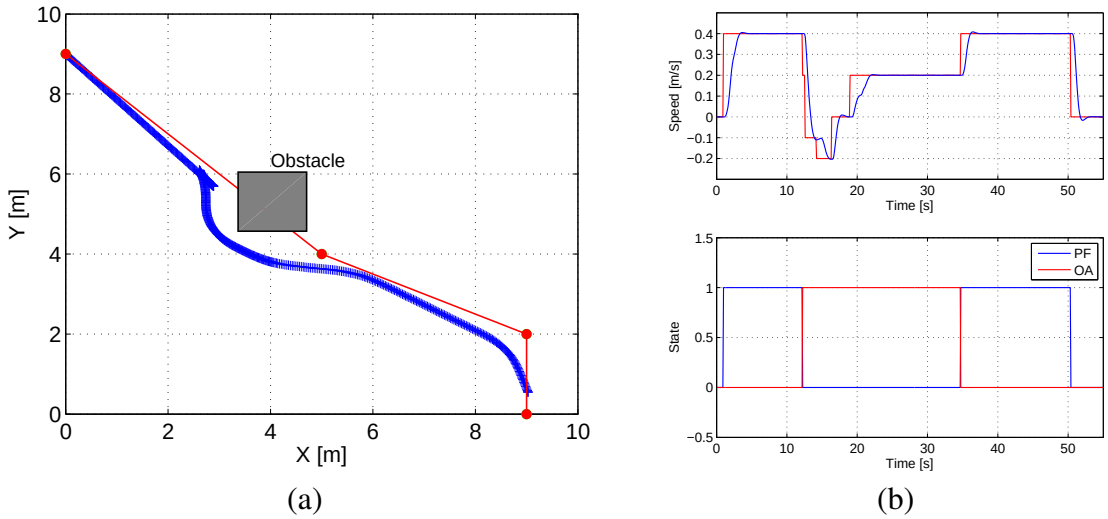


Figure 19: A target-assessment mission. A team of six robots is commanded to approach a target and relay camera images that allow the user to assess the target's condition. Initially, two scout robots inspect the path to ensure its safety. The four remaining robots then proceed in formation to the target and send back images.



(a) (b)

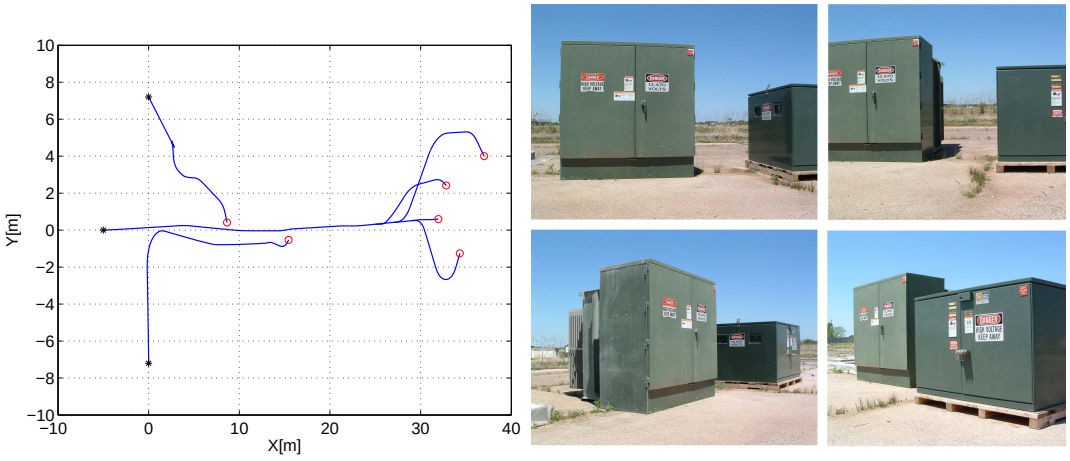
Figure 20: (a) Scout 1 in operation. (b) Graphical user interface control of Scout 1. The multivehicle graphical user interface is used to control the surveillance operation. This task is accomplished by sending waypoints to the vehicles. Using the camera view in the lower left hand corner of (b), the user verifies that the path is safe.



(a) (b)

Figure 21: (a) Scout 1 position plot. (b) Scout 1 translational velocity and state plots. The first scout vehicle encounters an obstacle in the path, avoids it successfully, and positions itself at the surveillance point. The commanded and actual translational speeds are shown. The state diagram shows when the system switches from goal seeking to obstacle avoidance.

Figure 22: The convoy proceeds. The lead robot travels autonomously through the preassigned waypoints. Using communication, the other robots follow the leader in a line. Each follower attempts to drive at a safe distance from its preassigned leader. The robots safely traverse the path in formation.



(a) (b)

Figure 23: (a) Trajectory of the team of mobile agents. (b) Target-assessment pictures. Following the paths shown, the team successfully assumes positions around the target. Images from the camera on each agent are shown. The user's target assessment is based on these images.