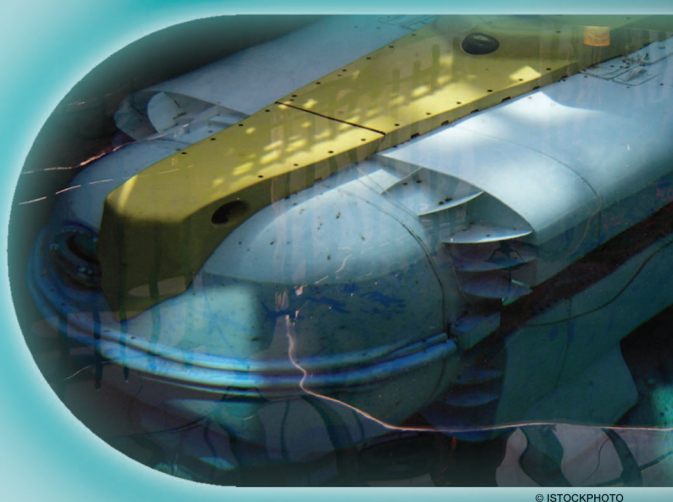
Experimental Testbed for Large Multirobot Teams

Verification and Validation

BY NATHAN MICHAEL, JONATHAN FINK, AND VIJAY KUMAR



Mobile Multirobot Systems

Experimental validation is particularly important in multirobot systems research. The differences between models and real-world conditions that may not be apparent in single robot experiments are amplified because of the large number of robots, interactions between robots, and the effects of asynchronous and distributed control, sensing, and actuation. Over the last two years, we have developed an experimental testbed to support research in multirobot systems with the goal of making it easy for users to model, design, benchmark, and validate algorithms. In this article, we describe our approach to the design of a large-scale multirobot system for the experimental verification and validation of a variety of distributed robotic applications in an indoor environment.

Our research focusses on decentralized multirobot algorithms that rely on an integrated approach to mobility, perception, and communication, with such applications as environmental monitoring, surveillance and reconnaissance for security and defense, and support for first responders in search and rescue operations [1]. In all of these applications, robots must rely on local sensing, computation, and control and exploit the availability of communication links with other robots whenever possible. To enable scaling up to large numbers, computations must be decentralized, and the system must be robust to changes in the numbers of robots and to the dynamic addition and deletion of units. There is also the need to provide some degree of centralization with an interface to one or more human operators for programming, tasking, and monitoring of the system.

These research applications serve as the motivation for our experimental testbed. While there is a rich body of work to build on, there is currently no inexpensive multirobot system that allows users to move easily from conceptual ideas to algorithms and then to experimentation. We begin by motivating design considerations for the testbed in the context of our research and existing multirobot control and experimental architectures. We next arrive at a set of design requirements for the system based on the driving applications as well as practical considerations. Most importantly, we are driven by the pragmatic considerations of ease of use, robustness, flexibility, and scalability to enable the easy inclusion of more robots and sensors with minimal changes to the existing infrastructure. We also review some of the applicable hardware and software options currently available. The experimental testbed is discussed in detail with overviews of the robots, software, and the supporting infrastructure required for multirobot experiments. Since simulation is of great relevance in the experimental process and the testbed design, we discuss its role and detail the transition from simulation to reality. Finally, we present several multirobot experiments for formation control and cooperative manipulation, which demonstrate the capabilities of the system for verification purposes and elucidate the experiment design process with our testbed.

Digital Object Identifier 10.1109/M-RA.2007.914924

1070-9932/08/$25.00©2008 IEEE

Motivating Design Considerations

A number of multirobot control and experimental architectures [2]–[4] have been developed over the years for use with teams of robots on the ground [5], [6], in the air [7], or under water [8], many of which were inspired by behavior-based control paradigms [9]. Often, architectures rely on hierarchy to manage the complexity of the task and the control software [10]. Additionally, the need to have decentralized control to enable scalability to large numbers is clear [3], [11]. However, to command large groups of robots, it is also essential to include an element of centralization to allow humans to interact and task the team.

The design of the experimental testbed was motivated by our interest in multirobot control for the deployment of potentially large numbers of cooperating robots with applications to tasks such as persistent surveillance, object manipulation, and transportation. We have proposed several methodologies in the context of these applications such as formation control [12]–[14], cooperative manipulation [15], and pattern generation [16] with the requirement that the algorithm adheres to three attributes: decentralization, anonymity, and uniform modularity. Decentralization means that the algorithm does not require access to the full global state and all control computations are done locally. Anonymity implies that the algorithm does not require robots to identify each other. Uniform modularity in algorithm implementation extends the idea of anonymity to further promote the notion that each robot executes an instance of the same uniform algorithm module. Modularity permits a higher level of interoperability between different control algorithms and often reduces the complexity of the control algorithm, thus simplifying the implementation. These attributes also improve the efficiency and interoperability of algorithms by permitting computations to execute in parallel across the robot network. Additionally, all robots are considered to be similar if not identical. The algorithm is made robust by ensuring that no single robot plays a role of vital importance or is unique in any way, and each robot is easily replaceable in the case of failure.

In light of these attributes, we advocate an asymmetric broadcast control (ABC) paradigm [17] in which all robots have identical software and receive identical instructions but have the intelligence in the software to differentiate, adopt roles, and perform the required tasks. One or more supervisory nodes serve to provide a degree of centralization by estimating partial global state information about the multirobot system. Such a paradigm is beneficial as we scale up to large numbers of robots, for example, numbers that are characteristic of sensor networks [18]. It becomes necessary to consider approaches to program, command, control, and monitor the robot teams without requiring knowledge of the specifics of the robots and the number of robots in the team. The asymmetry refers to the large volume of information that can be broadcast to the multirobot system relative to the partial state information sensed by or communicated back to the supervisory node.

System Requirements

The motivating design considerations and attributes discussed previously and the need to build a system that is adaptable to a range of multirobot applications lead to the following requirements:

- robust and reliable
- scalable and allows for the easy addition or deletion of agents
- capable of measuring and logging state information (including ground truth) for analysis
- extensible to a variety of applications
- inexpensive
- easy to use and maintain.

Robustness and reliability are of great concern when designing an experimental testbed. Since an assumption is made on the performance of the testbed when evaluating an algorithm, uncharacterized failure modes prevent accurate verification. Scalability is the focus of much of our research and cannot be limited by the system implementation. Measurements, state information, and algorithm status provide insight into the performance of the algorithms being tested and are invaluable during debugging. The ability to access or log such information at run time or for postprocessing is vital to the analysis of any experiment. Extensibility ensures that the testbed can be used to test a wide range of algorithms. By requiring that the system supports applications that demand significant computation, communication, and environmental sensing, we also enable the system to support algorithms that are less demanding but still require verification. With this requirement, we are also able to ensure that we support the many desirable properties previously discussed. The system must be designed to be inexpensive to allow researchers to incrementally increase the size of the system. Ease of use and maintenance is of great concern when the testbed consists of multiple independent units and supports collaborative research with many individuals accessing the system.

Resources for Multirobot Experimentation

Many resources currently exist for multirobot experimentation. We reviewed several hardware and software systems in the context of the system requirements discussed previously for suitability while designing the experimental testbed.

Hardware for Multirobot Experimentation

Robot selection is of crucial relevance when designing an experimental testbed. Since many robots may be used during an experiment, the capabilities, cost, and ease of maintenance are important considerations. The range of applicable algorithms is limited by the capabilities of the robot, particularly in distributed, decentralized, or sensor-rich algorithms, where the robots are expected to perform local computations and manage communication. The cost and ease of maintenance of the robots are relevant when the number of agents is increased or the hardware fails.

We considered many off-the-shelf platforms for indoor experimentation. The solutions we considered were often expensive, commercially unavailable, or did not lend themselves to multirobot experimentation. The three most promising designs were the SwarmBots from iRobot [19], the Khepera III from K-Team [20], and the ER1 from Evolution Robotics [21]. Unfortunately, the SwarmBot is not commercially available. The Khepera III was investigated but was found to have limited computational capabilities. Additionally,

the Khepera III requires familiarity with embedded Linux and software that support the necessary cross-compilation requirements. The ER1 was extensively tested but is no longer available as an individual unit. Indeed there were no commercially available mobile robots for less than US$5,000 with the computational capabilities of average laptops, sensors, and networking cards. Recently, iRobot has introduced the economical *Create* robot [22] which comes with actuation and a limited sensor suite. It is the most viable commercially available off-the-shelf solution at present. However, it lacks onboard processing and networking. For this reason, we chose to design and manufacture our custom robot.

Another important element of a multiagent testbed is a localization and ground-truth system. The system must be capable of estimating the pose of tens of robots simultaneously during an experiment. Applicable commercial systems are available including the Vicon MX System [23] from Vicon and Northstar [24] from Evolution Robotics. Although both of these systems were investigated, they were found to be either too expensive or impractical for our needs. Therefore, we developed a custom localization and tracking solution.

Support Software for Robotics

Software for even a single robot is a complex undertaking involving everything from low-level drivers for sensors and actuation up to higher-level computation and reasoning. For systems that integrate large numbers of agents, code modularity becomes even more important as one must also consider communications and networking between many agents. By writing drivers, controllers, and algorithms in a modular fashion, complex systems can be built that reside on a single agent or require the interaction of many modules on many agents.

Given adherence to writing and using modular, reusable code, it is inevitable that some pieces of even a highly customized multirobot system will already exist. This could range from a modern operating system to libraries that provide commonly used algorithms. An attribute by which most available software can be distinguished is licensing; i.e., distributed software is either open or closed source. When considering large teams of agents, the cost of licensing a proprietary operating system and other software can be significant.

Several open- and closed-source software libraries are available that support robotics and generally provide some or all of the following:

- an architecture with commonly defined interfaces so that software modules can be written that encourage good design practices and reuse
- a middleware library that allows both local and networked communication efficiently between modules
- a set of low-level drivers for robotic hardware
- a simulation environment to substitute when hardware is not necessary or available.

As such a system is extremely complex, most choose to not build a home-grown solution. Additionally, selecting an existing system with a large user-base and active development can lead to beneficial collaboration. There are a number of such systems that are currently available.

- *Microsoft Robotics Studio* [25]: Developed recently by Microsoft, this package provides a services-oriented architecture with both a visual programming environment and a physics-based simulator. It relies on proprietary modules to control and connect user-defined software modules in any language supported by Microsoft Visual Studio. This software dictates the use of a closed-source Windows operating system.
- *ORCA* [26]: This project leverages the separately developed Internet Communication Engine [27] middleware, which provides a host of features from a well-supported open-source project including easy interface definitions and tools to manage services, deployment, and event messaging. ORCA is released under the LGPL and GPL licenses and can be compiled on both Linux and Windows operating systems.
- *Open Robot Control Software* [28]: The OROCOS project has focused its development on real-time constraints that are often necessary in industrial robotics applications. OROCOS provides a component system using CORBA as a middleware as well as libraries for kinematics/dynamics and Bayesian filtering.
- *Player/Stage/Gazebo* [29]: Probably the most widely used robotics software package, the Player/Stage/Gazebo (PSG) project consists of libraries that provide access to communication and interface functionality. The robot server Player provides an architecture where many modules (known as drivers) can be independently written and connected through a custom middleware relying on transmission control protocol (TCP) communication. Users are also able to write simpler client applications that can connect to and command modules running on a Player server. Additionally, this project provides a two-dimensional simulator Stage and close collaboration with the three-dimensional physics-based simulation environment Gazebo. These simulators provide the powerful ability to transition transparently from code running on simulated hardware to real hardware. The project is developed for Unix-variant operating systems (e.g., Linux and Mac OS X).
- *Webots* [30]: A simulation environment for mobile robots relying on the open dynamics engine (ODE) [31] for physically accurate models, Webots has the capability of exporting control programs to a few select embedded robotic platforms. It is commercially available for multiple platforms (Windows, Linux, and Mac OS X) in a professional and less-enabled educational version.

We decided to pursue the open-source route, relying on the significant robotics user-base and the potential for growth in this area. Based on this decision, we chose to leverage the existing open-source software developed by the PSG project due to the availability of the three-dimensional physics-based simulation tools and the ability to write and test control software in simulation while moving seamlessly to experimentation with

hardware. We also find that this allows us to pursue collaborations with researchers who may not have access to our robots but are able to develop and test software in simulation with models of our robots.

Experimental Testbed Components

The experimental testbed consists of many components that are interfaced together to create the total system. In the discussion that follows, we present the robots, software, and infrastructure of the testbed.

Robots

As stated previously, we chose to design a robot for use in the testbed. The Scarab, a small differential drive robot, serves as the standard platform for multirobot experimentation. Additionally, we designed a cable robot platform, Khepri, which enables interaction with the team of robots as a global observer or aerial vehicle. The design and capabilities of each of these robots are detailed in the following sections.

Scarab Robot

As previously mentioned, we require a robot for indoor experimentation for algorithms that require local sensing, communication, and computation. Additionally, we wish to perform indoor experiments with large teams of robots with a limited experimental space. The robot must also be easily maintained, robust to failures, and economical.

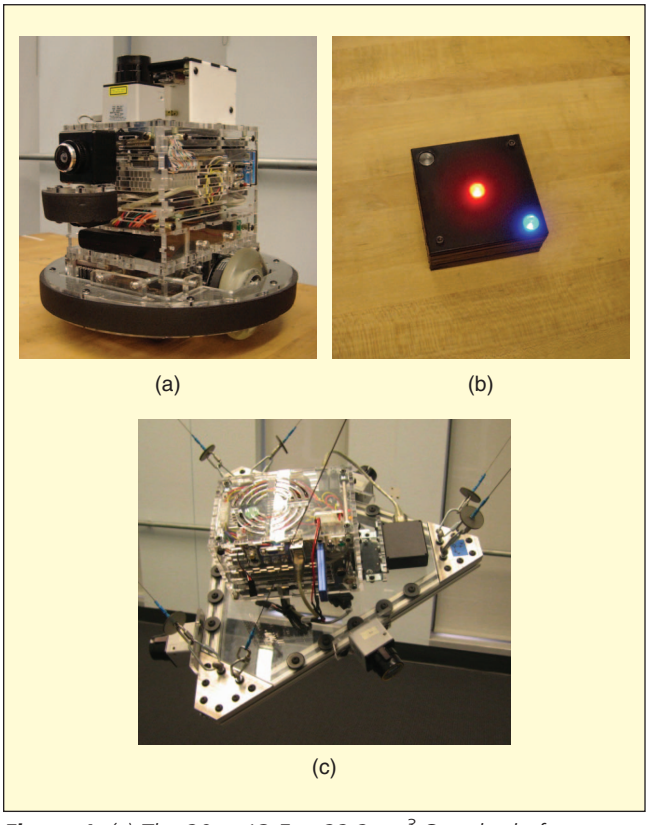


Figure 1. *(a) The 20 × 13.5 × 22.2 cm^3 Scarab platform. (b) An LED target is tracked for localization and ground truth on each of the robots. (c) The Khepri robot is controlled by six dc motors via pulleys and cables and has a full suite of sensing and computational abilities, making it well suited for emulation of an unmanned aerial vehicle in indoor environments.*

To achieve the above requirements, we developed the differential drive nonholonomic Scarab mobile robot shown in Figure 1(a). The design was completed using computer-aided design tools to be modular, easy to manufacture and assemble, and built from off-the-shelf components (see Figure 2).

Each Scarab is equipped with an onboard computer, power management system, wireless communication, and is actuated by stepper motors. The sensors, actuators, and controllers are modular and connected through the robotics bus [32] (which is derived from the controller area network protocol) or standard interfaces such as the universal serial bus or IEEE 1394. The result is a plug and play system where sensors and actuators can be added or removed from the hardware configuration.

The Scarab robot in Figure 1(a) depicts a typical platform configuration with a Hokuyo URG laser range finder and a Point Grey Firefly IEEE 1394 camera. This image also depicts the robot's foam bumper that protects the robot and allows it to interact with its environment. The physical dimensions of the robot in this configuration (less the bumper) are 20 × 13.5 × 22.2 cm^3 with a mass of 8 kg.

By designing the robot to be manufactured from readily available components and materials, the final cost of the robot shown in Figure 1(a) (without the camera or laser) is less than US$1,500. The end result is modular, easily maintained, and ready for application to a broad range of distributed robotics algorithms.

Khepri Robot

The experimental testbed also includes the Khepri, the aerial robot shown in Figure 1(c). Khepri is a six degree of freedom cable-controlled robot with the same onboard computing module as a Scarab. It is equipped with three Hokuyo URG laser range finders, a three axis inertial measurement unit, and a color Point Grey Dragonfly IEEE 1394 camera. The Khepri's kinematics and actuation system allow it to move in all six directions (positions and orientations), but the workspace is constrained since the cable tensions must be nonnegative [33].

By introducing the Khepri into the testbed, we are able to study interactions between the team of Scarabs and the Khepri

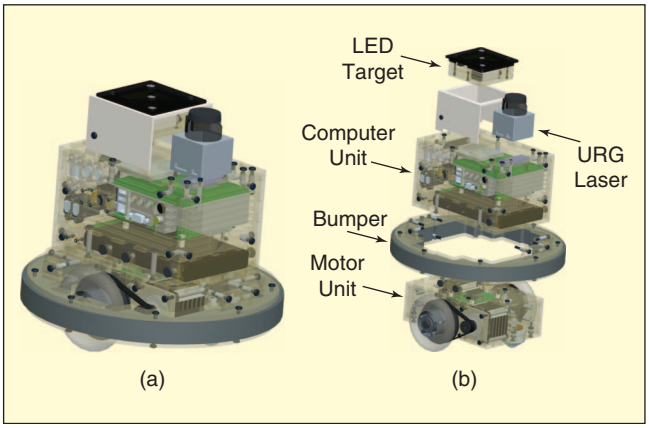


Figure 2. *Computer-aided design drawings showing the basic components of the Scarab and an exploded view of the robot design with relevant labels.*

and consider heterogeneous multirobot applications requiring a supervisor (as in the ABC paradigm). The distributed formation control discussed in the "Experimental Validation" section exemplifies an application requiring a supervisory agent with onboard sensing and computation capabilities.

Software

As discussed previously, we decided to use the open-source software developed by the PSG project. The choice of middleware is crucial in any multirobot testbed. It is the enabling factor that defines the networking and programming frameworks to which all algorithms must adhere or adapt. We have found that the capabilities provided by PSG are sufficiently flexible and transparent that most algorithms are easily accommodated to the framework design. As PSG is open source, modifications to the middleware are straightforward if new features are required.

Two methods exist for interfacing with the robots and sensors within the testbed via PSG (drivers and clients) using a variety of programming languages including C, C++, MATLAB, and Python. A driver is a code module that runs locally on the robot or computers in the testbed and is able to send and receive data to other drivers running locally or across the network. Such a design pattern permits the construction of code modules that run in their own thread and are able to manage both algorithm updates and communications with other robots and other local code modules. By ensuring that algorithms are properly programmed as drivers with strong interfacing, we are able to create identical reusable code modules for use on an individual robot, computer, or a large team of robots. The PSG client is an application that communicates with a driver but does not publish data accessible to other drivers. Generally in our system, clients serve as a simple way to interact with the robots.

Each experiment is defined by configuration files that are loaded by the Player server at runtime. These configuration files determine which code modules each robot or computer runs as well as the communication links required between the agents.

Since the system is distributed across many robots and computers, all information is generated and computed locally. However, a paradigm that requires global information can be implemented by writing a code that uses shared memory (often as a client). As our research interests pertain to distributed and decentralized algorithms, we generally write modules that operate asynchronously across the system without shared memory and with access only to information acquired locally or from network communications.

Infrastructure

Instrumentation for Localization and Ground Truth

We have developed a ground truth verification system consisting of a target with LED markers shown in Figure 1(b) and a network of overhead IEEE 1394 Point Grey Color Dragonfly cameras. Each marker contains three LEDs that flash an 8 B identification pattern that is detected and tracked by the overhead cameras to provide pose information. Measurements from multiple cameras are fused with an extended Kalman filter (EKF) to provide pose and uncertainty estimates for each robot in a global reference frame. For further refinement, an EKF runs on each of the robots, incorporating local odometry motion and the overhead tracking estimates.

The overhead tracking system allows control algorithms to assume pose is known in a global reference frame, thus eliminating the localization problem. Conversely, the tracking system allows the verification of localization algorithms as ground truth. It is also possible to use the tracking system in lieu of sensors that may be unavailable, such as neighbor sensors or collision avoidance sensors.

By the definition of the blinking pattern, the tracking system is theoretically capable of detecting 64 markers simultaneously. While the system has never been tested at its theoretical limit, it has been successfully used to track tens of robots simultaneously with a position error of approximately 2 cm and an orientation error of 5° at 29 Hz in a $9 \times 6 \times 6$ m^3 room. These values are based on raw data without any filtering either at the source or at the robot. While commercial tracking systems exist with higher accuracy [23], it should be noted that the cost difference between our system and commercial systems is significant. The tracking system consists of IEEE 1394 cameras, computers for image processing, and tracking targets that have a unit cost less than US$50.

Network

Since we need a low-latency network to communicate between agents and controllers with reasonable data rates, we use a dedicated 802.11a wireless network in a frequency range not used by adjacent wireless networks to ensure that we have complete control over the bandwidth available to the robots. We have successfully experimented with tens of computers, robots, and sensors performing data intensive experiments without a noticeable impact on the performance or latency of the network.

Data Logging

A requisite component of an experimental system is logging functionality. The system design permits local or networked data logging, depending on the demands of the experiment. Logging to local storage or mounted network drives on each robot is possible, depending on the space and the logging frequency required. Additionally, since we use PSG, a common logging interface exists that permits networked logging. As each robot communicates with other robots in the system, the same messages are sent to a computer that stores the data for postprocessing. With such a design, we are able to log relevant system information without requiring significant computational overhead from the robots.

Additional Considerations

The robots and the supporting computer infrastructure are networked with a dedicated local area network managed by a server with networked storage and a centralized user database. A user remotely accesses the robots in the same way they would access a desktop computer, and all working repositories and code are mounted via network drives. Since the robots and workstations all use the same x86 computer architecture, the same compiled binaries work on all platforms for easy

development. Deployment is simple since the same storage is available on both robots and workstations. By viewing the team of robots as a system of networked computers and using off-the-shelf technology, we are able to effortlessly distribute changes in the code base to all of the robots. Additionally, the dedicated server hosts web server capabilities, a repository for software and documentation, and other data to facilitate research and collaboration.

Simulation and Integration

As mentioned in the section on software, we use the software developed by the PSG project, which defines interfaces for our distributed system and provides communication between the robots. Additionally, Player provides a layer of hardware abstraction that permits algorithms to be tested in simulated three-dimensional Gazebo environments. Indeed all algorithm implementations and experiment designs (for example, those discussed in the "Formation Control" and "Cooperative Manipulation" sections) are identical for simulation and experimentation on hardware.

Gazebo incorporates dynamic interactions between models via ODE. Models of the environment of the local laboratory and hardware (discussed in the section on robots) have been reproduced in a simulated world (see Figures 3 and 4). The robot models accurately reflect the geometric, kinematic, and dynamic descriptions of the local robots used in the hardware implementation. Frictional coefficients and contact models (for environment interaction) have been estimated and incorporated into the simulations.

The robotics middleware (discussed previously) is the key to achieving seamless integration between components. The middleware offers clearly defined interfaces that carry out the following functions:

- permit software to be reused for multiple experiments
- allow new hardware or sensors to be rapidly introduced to the system
- enable tight integration between simulation and the real-world
- facilitate collaboration.

The third and fourth points emphasize the benefit of common middleware and interfaces. By defining a common interface structure, simulation environments (such as Stage and Gazebo) may be enabled to support the interfaces. This allows the code written for a simulation environment to be gracefully transitioned to the hardware. The same code that runs on a local computer in simulation will function in the same way on the robots. Additionally, software written using common robotics middleware allows for collaborations by requiring common interfaces between software.

By integrating the simulation environment into the testbed design and ensuring compatibility between the two, we are able to test both the algorithms and the experiment design. Since the same middleware and code base are used during simulation and experimentation, we are able to test the soundness of the experiment design and isolate possible points of failure or weakness that relate to issues not commonly addressed during algorithm verification, such as communication or memory constraints.

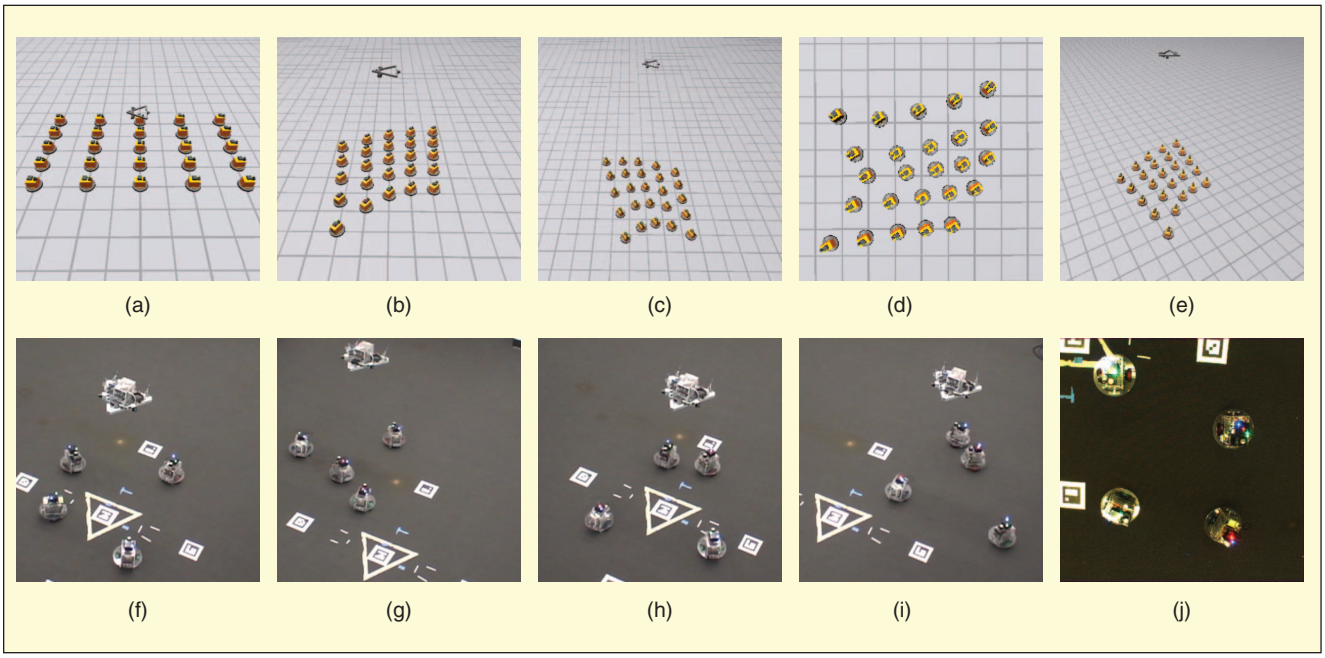


Figure 3. *Results from representative simulation and experimentation runs of formation control: (a)–(e) A representative trial run simulated in Gazebo. (a) The starting formation of twenty-five robots. (b) and (c) The motion of the group given a sinusoidal trajectory on the abstract manifold. (c) and (d) A snapshot of the robots and the corresponding view from the aerial robot's camera. (e) The final formation of the system. (f)–(j) Snapshots that are similar to (a)–(e) but with four Scarab robots. (f) The start configuration. (g) The convergence of the ground robots to* $a^{\text{des}} = \{1, 1, 0.5, 1, 0.5\}$ *(where* $a^{\text{des}} = \{\mu_x, \mu_y, \theta, s_1, s_2\}$*). (h) and (i) The motion of the system to* $a^{\text{des}} = \{1, -1, -0.5, 0.5, 1\}$*. (j) An image from the camera on the aerial robot. The Khepri controls to* $x = \mu_x, y = \mu_y,$ *and* $z = 3.0$ *m or* $z = 1.5$ *m in simulation and experimentation, respectively.*

There are occasions when simulation does not completely capture the behavior of the robots due to differences between reality and the simulated environment. These differences consist of model inaccuracies, simulation approximations, and local rather than distributed communication links. The difference between simulation and experimentation can be particularly significant in experiments involving physical contact between objects where models of frictional contact and the numerical methods for integration need to be more sophisticated than ODE for accurate prediction.

Experimental Validation

In the following discussion, we review recent results in distributed formation control and cooperative manipulation for a team of robots. The discussion emphasizes the implementation of these control algorithms using the experimental testbed.

Formation Control

Formation Control Algorithm

We are interested in controlling the shape, position, and orientation of a formation of a large team of nonholonomic ground robots in a decentralized manner using algorithms that are invariant to the number of ground robots. We briefly present experimental results using the Khepri aerial robot and a team of Scarab robots based on our previous work in [12]–[14]. The central idea is the development of an abstract description of the team of ground robots, which allows the aerial platform to control the team without any knowledge of the specifics of individual vehicles. The abstract description takes the form of a concentration or spanning ellipse defined by its pose ($\mu \in \mathbb{R}^2, \theta \in \mathbb{R}$) and shape ($s_1, s_2 \in \mathbb{R}$) along the major and minor axes. Thus, the pose and shape of the team of ground robots is a point on an abstract manifold. A controller on the abstract manifold yields changes in the abstract state necessary to drive the pose and shape of the formation to its desired value. Consistent with the ABC paradigm, the measured abstract state and the desired changes in the abstract state are broadcast to all of the Scarabs. Individual robot controllers with information about the abstract state (pose and shape) and their own local information ensure that the changes in the abstract state are achieved. Interagent collisions are resolved by constructing local control strategies that do not change the overall abstract state description [14].

We advocate an asymmetric broadcast control paradigm in which all robots have identical software and receive identical instructions.

Experimental Results and Ramifications

We experimentally validated the control law using the Khepri as a supervisory agent, which estimated the abstract state based on local observations from an onboard camera and performed the necessary computations required to control the abstract state. As seen in Figure 3(j), the Khepri is able to control the gross position and orientation of the formation as well as the shape by simply broadcasting the current observed abstract state, $\mathbf{a} = (\mu_x, \mu_y, \theta, s_1, s_2)$, and the desired abstract state, $\mathbf{a}^{des}$, to the ground robots. The Scarabs receive a broadcast abstract control command from the Khepri derived from its abstract state controller. Each Scarab locally computes its

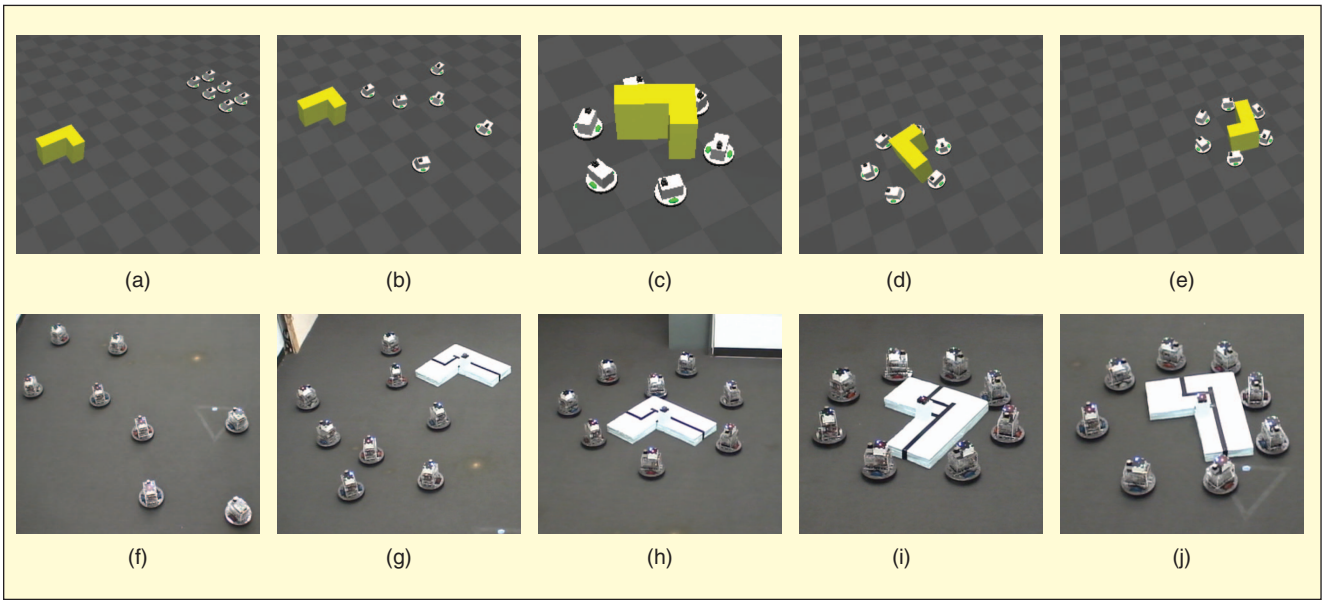


Figure 4. *Results from representative simulation and experimentation runs of cooperative manipulation. (a)–(e) The L-shaped cooperative manipulation in Gazebo. The robots all start using the approach controller [(a) and (b)], switch to the surround controller (c), and then to the transport controller [(d) and (e)], thus manipulating the object. (f)–(j) Similar snapshots: approach in (f) and (g), surround in (h), and transport in (i) and (j).*

The Scarab serves as the standard platform for multirobot experimentation.

own control inputs, which are velocities in the horizontal plane, based on this broadcast command as well as its current state and local neighbor measurements (for collision avoidance). A feedback-linearization scheme converts the linear velocities to forward and turning velocities for the nonholonomic Scarab.

These experiments highlight the importance of integrating simulation and experimentation during the implementation process. In simulation, we examined scenarios that required greater local computations and complexity by considering many more agents. We isolated points of fragility in the control algorithm and presented practical solutions to overcome these issues before working with the hardware [14].

Cooperative Manipulation

Cooperative Manipulation Algorithm

In a series of articles and papers, we described our approach to cooperative manipulation, which involves caging the manipulated object and moving while maintaining a condition of object closure [15], [34]. As in the previous subsection and consistent with the ABC paradigm, a geometric description of the manipulated object and the desired reference trajectory for manipulation is obtained by a supervisory agent and is broadcast to the team of Scarabs. Each robot chooses from a suite of controllers (vector fields) each of which is carefully constructed to guarantee properties of interest. For example, an approach controller guarantees that a robot will approach the object to be manipulated, while a surround controller ensures that a robot will go around the object and orbit it [16], [35], [36]. A transport controller allows each robot to move along the reference trajectory while ensuring that the condition of object closure (or caging) is maintained. All controllers guarantee that there will be no collisions. The complexity of the control problem is reduced to the problem of sequentially composing these controllers or vector fields [15]. Since these controllers or vector fields depend only on the object's position and geometric shape and the desired trajectory for the object, the resulting control computations are independent of the number of agents and only require the assumption that the number of agents is sufficiently large to surround the object for caging purposes. The control law is anonymous in that the identification of individual agents is unnecessary and the number of robots can change dynamically.

Experimental Results and Ramifications

While in theory the discrete protocols and continuous controllers are all guaranteed to work, the interaction between the discrete and continuous components and the fact that each robot operates asynchronously necessitates validation through simulation and experimentation. We demonstrated using Gazebo and the testbed that the sequential composition of the three behaviors, approach, surround and transport, which involves switches between these behaviors, is robust to both the type of object being manipulated and the number of robots available for manipulation. On real hardware, we have conducted tens of trials with four to eight robots manipulating an object along linear and sinusoidal trajectories as shown in Figure 4, as well as along trajectories obtained from a navigation function.

Through simulation and experimental trials, we demonstrated that the environment models in Gazebo mirrored reality to a sufficient degree that we returned to simulation and assessed large sets of initial conditions and parameters for testing and analysis. Such hardware and software integration lead to a significant speedup in the experimental process.

Conclusions

In this article, we presented our experimental testbed for a large team of robots and sensors, describing the hardware, software, and infrastructure for experimentation as well as the rationale for the design choices. In addition, we discussed our framework for developing software and some experimental results from recent studies. Our testbed enables us to validate distributed robotics algorithms for large numbers of robots engaged in a variety of tasks including formation control, search and pursuit of targets, and cooperative manipulation. This work also highlights a major benefit of selecting Player and Gazebo as an enabling mechanism to evaluate distributed robotics algorithms in simulation and on real robots. While our main focus in this article was on control algorithms, we intend to develop algorithms and software for distributed estimation and mapping from onboard sensors and look forward to reporting these advances in the future.

The application of multirobot theory to real-world scenarios requires the consideration of many challenging details that increase the complexity of implementation. It is clear that relaxing the assumptions of point models, Euclidean dynamics, and synchrony for multiagent systems is nontrivial. Further, multiagent systems require significant hardware, software, networking, and infrastructure support. To surmount these issues as multiagent systems scale in complexity and size, we advocate a close integration of high-fidelity simulation and experimentation and a carefully designed testbed that is constructed of robust, modular, and inexpensive components.

Acknowledgments

This research was supported by NSF grants CCR02-05336, NSF IIS-0413138; and IIS-0427313; ARO Grants W911NF-04-1-0148 and W911NF-05-1-0219; and ONR Grant N00014-07-1-0829.

Keywords

Multirobot systems, experimental robotics, decentralized control, formation control, cooperative manipulation.

References

[1] V. Kumar, D. Rus, and S. Singh, "Robot and sensor networks for first responders," *IEEE Pervasive Comput.*, vol. 3, no. 4, pp. 24–33, Oct. 2004.

[2] R. W. Beard, J. Lawton, and F. Y. Hadaegh, "A coordination architecture for spacecraft formation control," *IEEE Trans. Control Syst. Technol.*, vol. 9, no. 6, pp. 777–790, Nov. 2001.

[3] L. E. Parker, "Alliance: An architecture for fault tolerant multi-robot cooperation," *IEEE Trans. Robot. Automat.*, vol. 14, no. 2, pp. 220–240, Apr. 1998.

[4] A. Makarenko, A. Brooks, S. Williams, H. Durrant-Whyte, and B. Grocholsky, "A decentralized architecture for active sensor networks," in *Proc. IEEE Int. Conf. Robotics and Automation*, vol. 2, New Orleans, LA, Apr. 2004, pp. 1097–1102.

[5] R. Fierro, A. Das, J. Spletzer, J. Esposito, V. Kumar, J. P. Ostrowski, G. Pappas, C. J. Taylor, Y. Hur, R. Alur, I. Lee, G. Grudic, and B. Southall, "A framework and architecture for multi-robot coordination," *Int. J. Robot. Res.*, vol. 21, nos. 10–11, pp. 977–995, Oct. 2002.

[6] D. Cruz, J. Mcclintock, B. Perteet, O. A. A. Orqueda, Y. Cao, and R. Fierro, "Decentralized cooperative control—A multivehicle platform for research in network embedded systems," *IEEE Control Syst. Mag.*, vol. 27, no. 3, pp. 58–78, Jun. 2007.

[7] M. Valenti, B. Bethke, G. Fiore, and J. How, "Indoor multi-vehicle flight testbed for fault detection, isolation, and recovery," presented at the *AIAA Guidance, Navigation, and Control Conf. and Exhibit*, Keystone, CO, Aug. 2006.

[8] R. Bachmayer, N. E. Leonard, J. Graver, E. Fiorelli, P. Bhatta, and D. Paley, "Underwater gliders: Recent developments and future applications," in *Proc. Int. Symp. Underwater Technology*, Taipei, Taiwan, Apr. 2004, pp. 195–200.

[9] R. C. Arkin, *Behavior Based Robotics*. Cambridge, MA: MIT Press, 1998.

[10] R. G. Simmons, S. Singh, D. Hershberger, J. Ramos, and T. Smith, *First Results in the Coordination of Heterogeneous Robots for Large-Scale Assembly* (Lecture Notes Series in Control and Information Sciences). New York: Springer-Verlag, 2000, vol. 271, pp. 323–332.

[11] J. T. Feddema and D. Schoenwald, "Decentralized control of cooperative robotic vehicles," *Proc. SPIE*, vol. 4364, pp. 136–146, Sept. 2001.

[12] C. Belta and V. Kumar, "Abstraction and control for groups of robots," *IEEE Trans. Robot*, vol. 20, no. 5, pp. 865–875, Oct. 2004.

[13] N. Michael, C. Belta, and V. Kumar, "Controlling three-dimensional swarms of robots," in *IEEE Int. Conf. Robotics and Automation*, Orlando, FL, May 2006, pp. 964–969.

[14] N. Michael, J. Fink, and V. Kumar, "Controlling a team of ground robots via an aerial robot," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, San Diego, CA, Oct. 2007, pp. 965–970.

[15] J. Fink, N. Michael, and V. Kumar, "Composition of vector fields for multirobot manipulation via caging," presented at *Robotics: Science and Systems*, Atlanta, GA, Jun. 2007.

[16] M. A. Hsieh and V. Kumar, "Pattern generation with multiple robots," in *Proc. IEEE Int. Conf. Robotics and Automation*, Orlando, FL, May 2006, pp. 2442–2447.

[17] N. Michael, J. Fink, S. G. Loizou, and V. Kumar, "Architecture, abstractions, and algorithms for controlling large teams of robots: Experimental testbed and results," presented at *Int. Symp. Robotics Research*, Hiroshima, Japan, Nov. 2007.

[18] K. Whitehouse, "The design of calamari: An ad hoc localization system for sensor networks," M.S. thesis, Dept. Elect. Eng. Comp. Sci., Univ. California, Berkeley, 2002.

[19] J. McLurkin, "Stupid robot tricks: A behavior-based distributed algorithm library for programming swarms of robots," M.S. thesis, Dept. Elect. Eng. Comp. Sci., MIT, Cambridge, MA, 2004.

[20] K-Team. [Online]. Available: http://www.k-team.com

[21] ER1 Personal Robot System. [Online]. Available: http://www.evolution.com/er1

[22] iRobot Create Programmable Robot. [Online]. Available: http://www.irobot.com/create

[23] Vicon MX Systems. [Online]. Available: http://www.vicon.com/products/viconmx.html

[24] NorthStar. [Online]. Available: http://www.evolution.com/products/northstar

[25] Microsoft Robotics Studio SDK. [Online]. Available: http://msdn.microsoft.com/robotics

[26] A. Brooks, T. Kaupp, A. Makarenko, S. B. Williams, and A. Oreback, *Software Engineering for Experimental Robotics* (Springer Tracts Series in Advanced Robotics). Berlin, Germany: Springer-Verlag, 2007, vol. 30, pp. 231–251.

[27] Internet Communications Engine. [Online]. Available: http://www.zeroc.com/ice.html

[28] H. Bruyninckx, "Open robot control software: The OROCOS project," in *Proc. IEEE Int. Conf. Robotics and Automation*, Seoul, Korea, May 2001, vol. 3, pp. 2523–2528.

[29] B. P. Gerkey, R. T. Vaughan, and A. Howard, "The player/stage project: Tools for multi-robot and distributed sensor systems," in *Proc. Int. Conf. Advanced Robotics*, Coimbra, Portugal, Jun. 2003, pp. 317–323.

[30] Webots, *Fast prototyping and simulation of mobile robots*. [Online]. Available: http://www.cyberbotics.com/products/webots

[31] Open Dynamics Engine. [Online]. Available: http://www.ode.org

[32] D. Gomez-Ibanez, E. Stump, B. Grocholsky, V. Kumar, and C. J. Taylor, "The Robotics Bus: A local communcations bus for robots," *Proc. SPIE*, Ser. XVII, vol. 5609, pp. 155–163, Dec. 2004.

[33] E. Stump and V. Kumar, "Workspaces of cable-actuated parallel manipulators," *ASME J. Mech. Des*, vol. 128, no. 1, pp. 159–167, Jan. 2006.

[34] G. A. S. Pereira, M. F. M. Campos, and V. Kumar, "Decentralized algorithms for multi-robot manipulation via caging," *Int. J. Robot. Res.*, vol. 23, nos. 7–8, pp. 783–795, July 2004.

[35] L. Chaimowicz, N. Michael, and V. Kumar, "Controlling swarms of robots using interpolated implicit functions," in *Proc. IEEE Int. Conf. Robotics and Automation*, Barcelona, Spain, Apr. 2005, pp. 2487–2492.

[36] M. A. Hsieh, S. G. Loizou, and V. Kumar, "Stabilization of multiple robots on stable orbits via local sensing," in *Proc. IEEE Int. Conf. Robotics and Automation*, Rome, Italy, Apr. 2007, pp. 2312–2317.

Nathan Michael is currently pursuing a Ph.D. degree in mechanical engineering at the University of Pennsylvania, Philadelphia. His research interests include control and estimation for multirobot systems and experimental robotics.

Jonathan Fink received a dual B.S. degree in electrical and computer systems engineering at Rennselaer Polytechnic Institute, New York, in May 2004. He is currently pursuing a Ph.D. degree in electrical and systems engineering at the University of Pennsylvania, Philadelphia. He is a member of the GRASP Lab of the University of Pennsylvania. His research interests include algorithms for multirobot systems with sensor network and manipulation applications.

Vijay Kumar received his M.Sc. and Ph.D. degrees in mechanical engineering from The Ohio State University, Columbus, in 1985 and 1987, respectively. He has been on the faculty in the Department of Mechanical Engineering and Applied Mechanics with a secondary appointment in the Department of Computer and Information Science at the University of Pennsylvania, Philadelphia, since 1987. He is currently the UPS Foundation professor and the chairman of mechanical engineering and applied mechanics.

Address for Correspondence: Nathan Michael, University of Pennsylvania, Philadelphia, PA 19104-6228, USA. E-mail: nmichael@grasp.upenn.edu.