A Framework and Architecture for Multi-Robot Coordination

Rafael Fierro

MARHES Laboratory
School of Electrical and Computer Engineering
Oklahoma State University
Stillwater, OK USA

Aveek Das
John Spletzer
Joel Esposito
Vijay Kumar
James P. Ostrowski
George Pappas
Camillo J. Taylor

GRASP Laboratory
University of Pennsylvania
Philadelphia, PA USA

Yerang Hur
Rajeev Alur
Insup Lee

SDRL Laboratory
University of Pennsylvania
Philadelphia, PA USA

Greg Grudic

Department of Computer Science
University of Colorado at Boulder
Boulder, CO USA

Ben Southall

Sarnoff Corporation
Princeton, USA

Abstract

In this paper, we present a framework and the software architecture for the deployment of multiple autonomous robots in an unstructured and unknown environment, with applications ranging from scouting and reconnaissance, to search and rescue, to manipulation tasks, to cooperative localization and mapping, and formation control. Our software framework allows a modular and hierarchical approach to programming deliberative and reactive behaviors in autonomous operation. Formal definitions for sequential composition, hierarchical composition, and parallel composition allow the bottom-up development of complex software systems. We demonstrate the algorithms and software on an experimental testbed that involves a group of car-like robots, each using a single omnidirectional camera as a sensor without explicit use of odometry.

KEY WORDS—multi-robot coordination, hierarchical hybrid systems, vision-based control

The International Journal of Robotics Research
Vol. 21, No. 10–11, October-November 2002, pp. 977-995,

1. Introduction

It has long been recognized that there are several tasks that can be performed more efficiently and robustly using multiple robots (Donald, Gariepy, and Rus 2000; Khatib et al. 1996). In fact, there is extensive literature on control and coordina-

tion for multiple mobile robots, and application to tasks such as exploration (Burgard et al. 2000), surveillance (Feddema and Schoenwald 2001), search and rescue (Jennings, Whelan, and Evans 1997), mapping of unknown or partially known environments (Taylor 2002), distributed manipulation (Rus, Donald, and Jennings 1995; Matarić, Nilsson, and Simsarian 1995), distributed sensor fusion and localization (Stroupe, Martin, and Balch 2001; Roumeliotis and Bekey 2000), and transportation of large objects (Stilwell and Bay 1993; Sugar and Kumar 2000; Kosuge et al. 1999). See, for instance, Parker (2000) for a review of contemporary work in this area.

In our previous work, a high-level language CHARON was introduced for describing hierarchical hybrid systems (Alur et al. 2000a). The problem of controlling a group of robots has been addressed in a series of papers: trajectory generation is addressed in Fierro et al. (2002) and Belta and Kumar (2001); switched control in Desai, Kumar, and Ostrowski (1999) and Fierro et al. (2001b); vision-based formation control is presented in Das et al. (2002); and cooperative localization and manipulation is addressed in Spletzer et al. (2001). In this paper, in contrast, we focus on the design of software and control behaviors for cooperative multi-robot systems. Our goal is to describe a set of tools that allows the development of controllers and estimators for multi-robot coordination. The tools consist of a framework for developing software components, an architecture for control and estimation modules, and a set of decentralized control, planning and sensing algorithms. In our software framework, each component, for example each robot, is an *agent*. The agent can be a parallel composition of many sub-agents, for example, sensor agents, actuator agents, and other software agents that operate in parallel. The multi-robot control task is decomposed within each agent into a set of *modes* or behaviors. Modes can consist of high-level behaviors such as planning a path to a goal position, as well as low-level tasks such as obstacle avoidance. We allow for sequential composition of modes to enable changes in behavior. Hierarchical composition allows high-level deliberative controllers to be composed with low-level reactive controllers. We use a high-level language to formally describe how and when transitions between these modes are to take place in order to achieve a set of global objectives.

2. Motivation

There is extensive literature on the control of robot manipulators or mobile robots in structured environments, and robot control is a well-understood problem area. However, traditional control theory mostly enables the design of controllers in a single mode of operation, in which the task and the model of the system are fixed. A similar problem exists in developing estimators in the context of sensing.

When operating in unstructured or dynamic environments with many different sources of uncertainty, it is very difficult if not impossible to design controllers that will guarantee performance even in a local sense. In contrast, we also know that it is relatively easy to design reactive controllers or behaviors that react to simple stimuli or commands from the environment. This is the basis for the subsumption architecture (Brooks 1986) and the paradigm for behavior-based robotics (Matarić 1995; Arkin 1998).

Our goal in this paper is to establish a paradigm that allows us to design simple components whose performance can be analyzed and predicted using control theory and dynamics, and to develop tools that allow us to construct hierarchical systems with switches in behavior that can be used in the development of intelligent robotic systems. Specifically, we describe an architecture and a high-level language with formal semantics, CHARON, that can be used to describe multi-agent, networked robotic systems with multiple control and estimation modes, and discrete communication protocols in a principled way. The architecture allows the development of complex multi-robot behavior via hierarchical and sequential composition of control and estimation *modes*, and parallel composition of *agents*. We present our ongoing work to automatically generate control and simulation code from the high-level language description.

We also illustrate the application of these ideas to the development of an experimental platform of multiple mobile robots that cooperate in performing the following tasks: (a) searching and identification of colored objects; (b) cooperative localization of targets and robots; (c) cooperative two-dimensional mapping; (d) cooperative manipulation and transportation of objects; and (e) formation control.

The experimental results demonstrate the benefits and the limitations of mode switching and the methodology underlying the implementation of cooperative control of multi-robotic systems.

3. Modeling Language and Software Architecture

The last few years have seen active research in the field of distributed robotics, and in the development of architectures for multi-robot coordination. These architectures have focused on providing different capabilities to the group of robots. For instance, ALLIANCE (Parker 1998), a behavior-based software architecture, has focused on fault tolerant cooperative control. In Morrow and Khosla (1997), robot skills are expressed as finite state machines (FSMs) under the *Chimera* software environment. The coordination of robots for large-scale assembly has been considered in Simmons et al. (2000). Klavins and Koditschek (2000) have presented tools for composing hybrid control programs for a class of distributed robotic systems. This approach assumes that a palette of controllers for individual tasks is available. These controllers i.e., robot behaviors are sequentially composed using the techniques introduced in Burridge, Rizzi, and Koditschek (1999). These

ideas are applied to the design of assembly tasks as found in automated factories.

The *three-tier* (3T) layered architecture is presented in Schreckenghost et al. (1998). In this work the problem of managing life support for remote facilities is considered. A planner coordinates the tasks across subsystems. An explicit separation between deliberative and reactive tasks enables appropriate human intervention in autonomous operation. Recently, the *CLARAty* architecture for robotic autonomy was introduced in Volpe et al. (2001). This two-tiered approach considers a tight coupling between planning and execution within the *decision layer*. The decision layer interacts with the *functional layer*. The functional layer consists of software modules for estimation, status reporting and system operation organized in a conventional object-oriented manner. Both architectures, 3T and *CLARAty*, are being implemented on some NASA robotic projects.

Our software architecture has some similarities with the works described above. It is object-oriented and supports hierarchical composition of agents and behaviors or modes. In addition, we use the theory of *hybrid systems* (Alur et al. 2000b; Fierro and Lewis 1997; van der Schaft and Schumacher 2000) to formally analyze and design multi-robotic cooperative systems. For this purpose, we have developed CHARON, an acronym for Coordinated Control, Hierarchical Design, Analysis, and Run-Time Monitoring of Hybrid Systems.

3.1. Modeling Language

CHARON is a language for modular specification of interacting hybrid systems based on the notions of agents and modes. For a hierarchical description of the system architecture, CHARON provides the operations of instantiation, hiding, and parallel composition on agents, which can be used to build a complex agent from other agents. The discrete and continuous behaviors of an agent are described using modes. For a hierarchical description of the behavior of an agent, CHARON supports the operations of instantiation and nesting of modes. Furthermore, features such as weak pre-emption, history retention, and externally defined Java functions, facilitate the description of complex discrete behavior. Continuous behavior can be specified using differential as well as algebraic constraints, and invariants restricting the flow spaces, all of which can be declared at various levels of the hierarchy. The modular structure of the language is not merely syntactic, but is also reflected in the semantics so that it can be exploited during analysis. The key features of CHARON are summarized below.

- **Architectural hierarchy:** The building block for describing the system architecture is an *agent* that communicates with its environment via shared variables and also communication channels. The language supports the operations of composition of agents for concurrency, *hiding* of variables for information encapsulation, and *instantiation* of agents to support reuse.
- **Behavioral hierarchy:** The building block for describing a flow of control inside an atomic agent is a *mode*. A mode is basically a hierarchical state machine, that is, a mode can have submodes and transitions connecting them. Variables can be declared locally inside any mode with standard scoping rules for visibility. Modes can be connected to each other through entry and exit points. We allow the instantiation of modes so that the same mode definition can be reused in multiple contexts. Finally, to support *exceptions*, the language allows group transitions from default exit points that are applicable to all enclosing modes, and to support *history retention*, the language allows default entry transitions that restore the local state within a mode from the most recent exit.
- **Discrete and continuous variable updates:** Discrete updates are specified by *guards* labeling transitions connecting the modes. Such updates correspond to mode-switching, and are allowed to modify variables through assignment statements.

  Variables in CHARON can be declared *analog*, and they flow continuously during the continuous updates that model the passage of time. The evolution of analog variables can be constrained in three ways: *differential* constraints (e.g., by equations such as $\dot{x} = f(x, u)$), *algebraic* constraints (e.g., by equations such as $y = g(x, u)$), and *invariants* (e.g., $x - y < c$) which limit the allowed durations of flows. Such constraints can be declared at different levels of the mode hierarchy.

It should be noted that CHARON is a *modeling language*: it supports nondeterminism for both discrete and continuous updates, it is suitable for describing the system as well as the assumptions about the environment in which the system is supposed to operate, and for describing the same system at different levels of abstraction. The language constructs primarily facilitate the description of control flow, but it also supports calls to externally defined Java functions which can be used to write complex data manipulations. More details about the language, the global semantics and the formal description are presented in Alur et al. (2000a).

3.2. Software Architecture

3.2.1. Architectural Modeling with Agents

The architecture proposed here allows the development of complex multi-robot behavior via hierarchical and sequential composition of control and estimation modes, and parallel composition of agents.[1]

1. Note that our definitions of composition do not satisfy the constraints required by Burridge, Rizzi, and Koditschek (1999), where the definition of composition comes with guarantees for global performance.

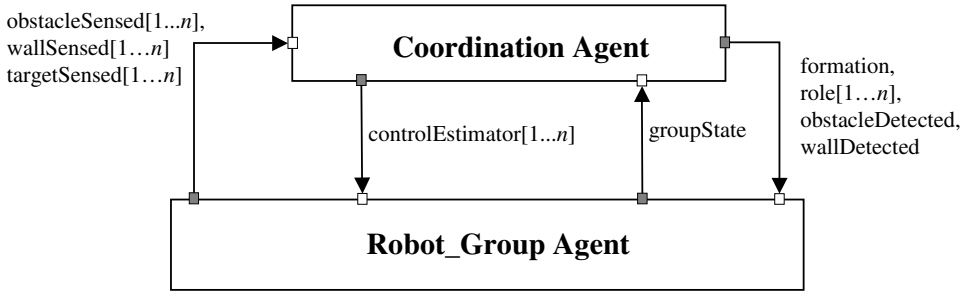

Fig. 1. Agent hierarchy diagram.

At the highest level of the hierarchy, the multi-robot system is represented by two interacting agents: a *coordination* agent and a *robot-group* agent. The coordination agent reduces to the specification of communication channels between robot agents, and the specification of parameters for transitions and the instantiation of each agent within the robot-group agent. This is schematically illustrated in Figure 1.

Robot agents can receive estimates of the obstacles from other robots, and commands and specifications from the human operator on input channels, and it can send its own information to other robots or to the human operator on the output channels (see Figure 2).

The architecture diagram in Figure 3 shows a parallel composition of control and estimator agents operating concurrently with off-the-shelf black boxes (shown shaded). The *control* agent switches between modes described in the next subsection. The estimator agent on the left in the dotted box represents different logical sensors that can be developed from one vision system.

3.2.2. Behavioral Modeling with Modes

The state of a robot agent is given by $\boldsymbol{x} \in \mathbb{R}^n$. Its evolution is determined by a set of differential equations

$$\dot{\boldsymbol{x}} = f_q(\boldsymbol{x}, \boldsymbol{u}), \qquad \boldsymbol{u} = k_q(\boldsymbol{x}, \boldsymbol{z}), \tag{1}$$

where $\boldsymbol{u} \in \mathbb{R}^m$ is the control vector, $q \in \mathbb{Q} \subset \mathbb{Z}$ is the control mode for the agent, $\mathbb{Q}$ is a finite set of control mode indices, $\mathbb{Z}$ denotes the set of positive integers, and $\boldsymbol{z} \in \mathbb{R}^p$ is the information about the external world available either through sensors or through communication channels. The robot agent contains modes describing behaviors that are available to the robot.

The *controller-Top* mode depicted in Figure 4 consists of two submodes: *leader* mode and *follower* mode. These submodes become active based on the state of the discrete variable *role* from the coordination agent.

There are three submodes within the leader mode: *goToGoal*, *obstacleAvoidance*, and *wallFollowing*. Initially, the leader's mode is *goToGoal* and both *wallDetected* and *obstacleAvoided* are false. If one of these becomes true, the transition from *goToGoal* to one of *wallFollowing* and *obstacleAvoidance* occurs accordingly, and the reverse transition will be enabled if the variable is reset. If both variables become true, *obstacleAvoidance* mode will be active. A leader will enter *wallFollowing* mode if the Boolean variable *wallDetected* becomes true, *obstacleAvoidance mode* if *obstacleDetected* becomes true, respectively. This behavioral structure is illustrated in Figure 5.

If a robot agent is in *follower* mode, it will follow its leader keeping a desired distance and relative bearing. The *estimator* agent provides all the required information about the state of the leader. The follower robot uses this information in order to compute its own control velocities. We have developed and implemented a number of controllers for this purpose. Figure 6 illustrates the textual description in CHARON of the *follower* mode. The separation-bearing controller (SBC) implemented within this mode is presented in Section 5.

In the next section, we proceed to illustrate how to exploit the modular structure of CHARON in implementing the above architecture. We consider the problem of controlling multiple mobile, autonomous robots for mission-critical applications and stringent requirements on safety. A detailed CHARON code for a two-robot example is given in the Appendix.

4. Real-Time Framework for Multi-Robot Coordination

Our multi-threaded software implementation encapsulates algorithms and data in the usual object-oriented manner together with control of a thread within which the algorithms will execute, and a number of events that allow communication with other objects. At the top of the hierarchy, the algorithms associated with the objects are likely to be planners, while at bottom they will be interfaces to control and sensing hardware. The planner objects are able to control the execution of the lower level objects to service high-level goals. To offer platform independence, only the lowest level objects should be specific to any hardware, and these should have

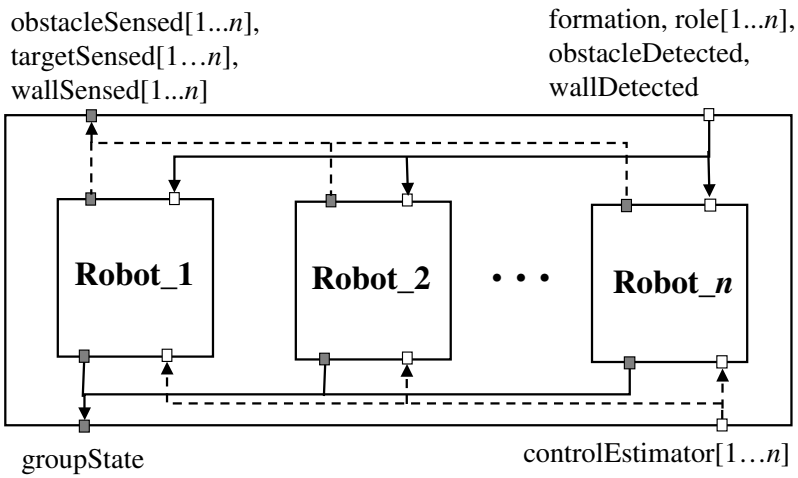

Fig. 2. Robot-group agent.

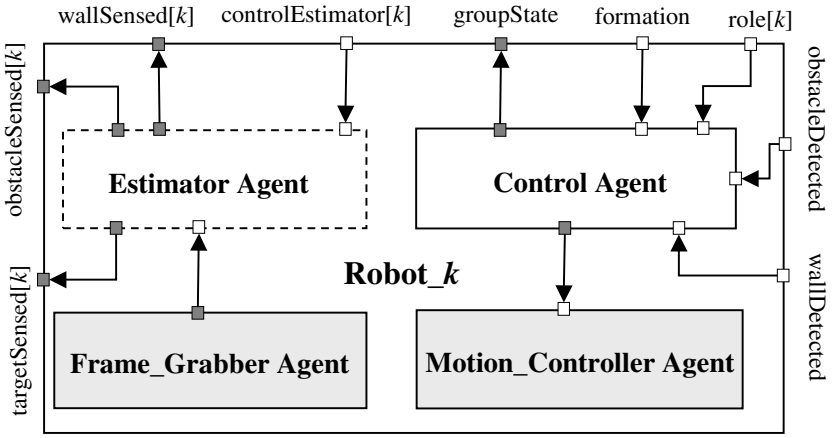

Fig. 3. A robot agent consists of estimator agent, control agent and hardware interface agents.

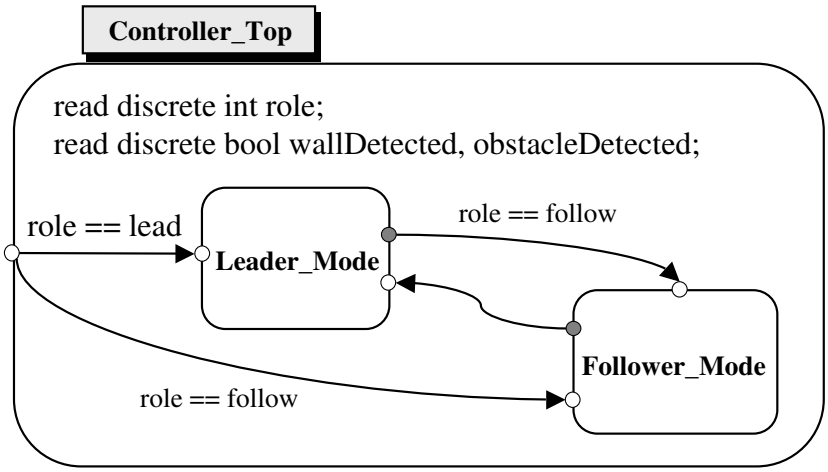

Fig. 4. Robot modes within the *Controller Top* mode.

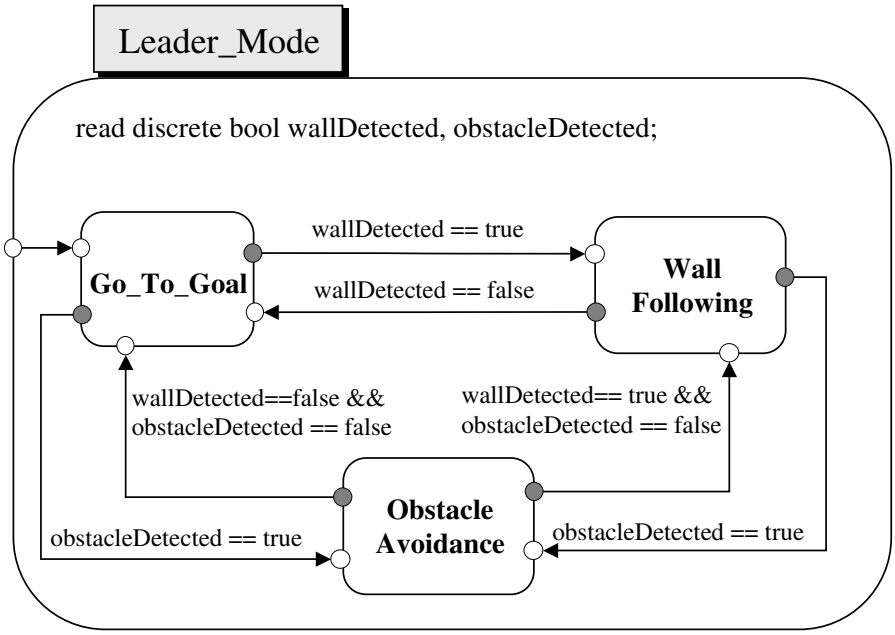

Fig. 5. Submodes within the *leader* mode.

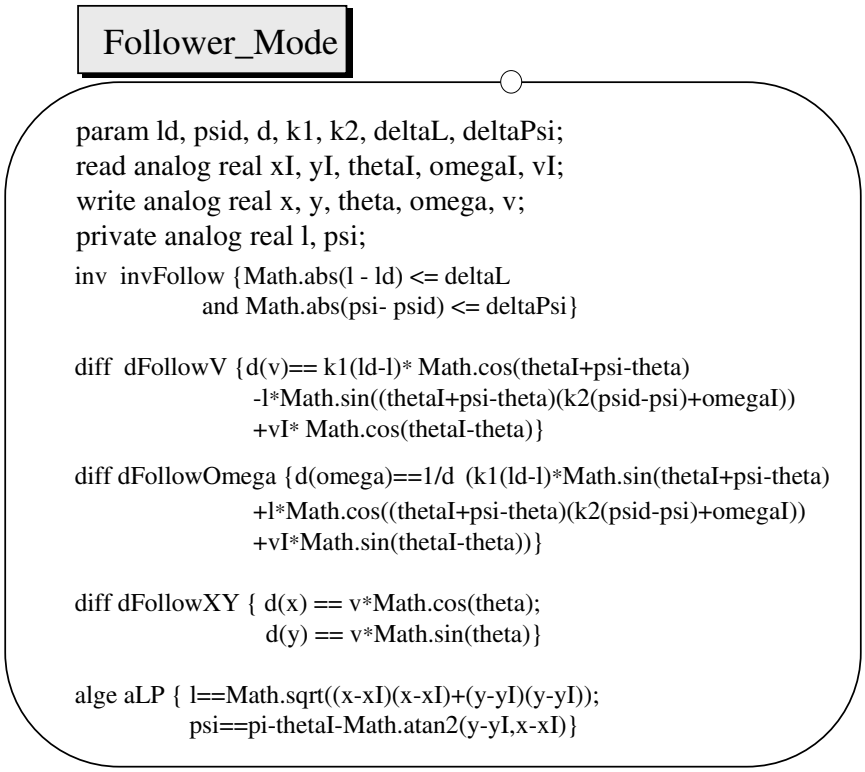

Fig. 6. Description of the *follower* mode.

a consistent interface for communication with the planning objects that control their execution.

Vision-based control algorithms have been incorporated into the multi-threaded software architecture for such basic functionality as obstacle avoidance, wall following, mapping and localization, and group behavior. Thus, each robot can be easily programmed to follow a wall while avoiding obstacles, and looking for targets that might specify the goal location. With multiple robots, each robot may also have the flexibility of following other robots. This software is used by robots to explore long passages in buildings and build maps of the environment for possible reconstruction with applications to immersion or for scouting and reconnaissance. Other examples of group behavior include *formation keeping*, *collaborative mapping and manipulation*. We can form two-robot or n-robot teams by parallel composition of robot agents. The availability and sharing of information between the robots allows us: (a) to design modes within *estimator* agents that can exploit sensory information obtained from other robots; and (b) to design the *coordination* agent to initiate or trigger mode-switching within the *controller* agent.

In the next subsections, we describe the real-time implementation of the software objects that realize the architecture proposed here.

4.1. Experimental Platform

The GRASP Laboratory Clodbuster (CB) robots served as the testbed for all experiments. The platform is based upon a commercially available radio control truck from Tamiya Inc., with significant modifications. The CB version used for this work lacks on-board processing. Wireless video is transmitted at 2.4 GHz, back to a remote computer where all vision and control algorithms are processed. Control servo signals are then sent back in turn from the computer to the CB via a parallel port interface. An image of the CB platform is shown in Figure 7 (Extension 1).

The CB platform uses an omnidirectional camera (*Paracamera* from Remote Reality) as its sole sensor. One of the primary advantages of these catadioptric sensors is that they afford a single effective point of projection (Baker and Nayar 1998). This means that after an appropriate calibration, every point in the omnidirectional image can be associated with a unique ray through the focal point of the camera. This allows azimuth and elevation angles to every teammate (and target) visible in the 360° *field-of-view* image to be estimated, making the camera an ideal choice for cooperative sensing tasks, as will be discussed in the following section.

The robot has a servo controller on board for steering and a digital proportional speed controller for forward/backward motion. A parallel port interface, also designed in our lab, allows us to drive up to eight mobile robot platforms from a single Windows NT workstation. A video receiver, located at the host computer, feeds the signal to a frame grabber that is able to capture video at full frame rate (30 Hz) for image processing. This yields a video signal in a format for viewing and recording, as well as image processing.

4.2. Sensors

Sensors are organized hierarchically within the *estimator* agent shown in Figure 3. The estimator agent will share information with the *coordination* and *robot* agents. The logical sensors or detectors work in parallel and all use the information provided by the *Frame_Grabber* agent. This is graphically depicted in Figure 8.

A target or other robots can be identified in the image using a YUV color space based feature extractor, which provides robustness to variations in illumination (Spletzer et al. 2001). Three-dimensional color models are generated a priori from images of the target at numerous distances, orientations, and illumination levels. These data are stored in a pair of look-up tables to speed image processing. During operation, the target detection algorithm, the *blobExtractor* sensor, is initially applied to the entire image and can run at frame rate (30 Hz).

Once the target is acquired, the sensor switches to *target tracking mode* (Extension 2). The target tracking scheme is simple yet robust. The YUV-based color extractor actually processes the entire image very quickly, segmenting up to eight colors from each pixel with a single binary operation.

4.2.1. Range Mapping

The range map (see Figure 7) is obtained by applying a Sobel gradient to the omnidirectional image. The resulting edges in the image are the features of interest. By assuming a ground plane constraint, the distance to the nearest feature in the sector of interest is determined from the its relative elevation angle to the mirror. This provides a range map to all obstacles at frame rate.

4.2.2. Localizer

Our localization algorithm employs an extended Kalman filter (EKF) to match landmark observations to an a priori map of landmark locations. The *Localizer* object uses the *blobExtractor sensor agent* to determine the range and the bearing of an observed landmark. If the observed landmark is successfully matched, it is used to update the vehicle position and orientation. The lower left image of Figure 7 shows a typical image used for localization.

In the experiment depicted in Figure 9, we let the robot trace an open-loop circular trajectory in a measured area with fixed landmarks. The robot usually sees very few landmarks. The overhead camera gives us an idea of the actual ground trajectory of the robot. The average error is approximately 2 cm. This is particularly challenging as we use a simplified kinematic model, and we lack odometry.

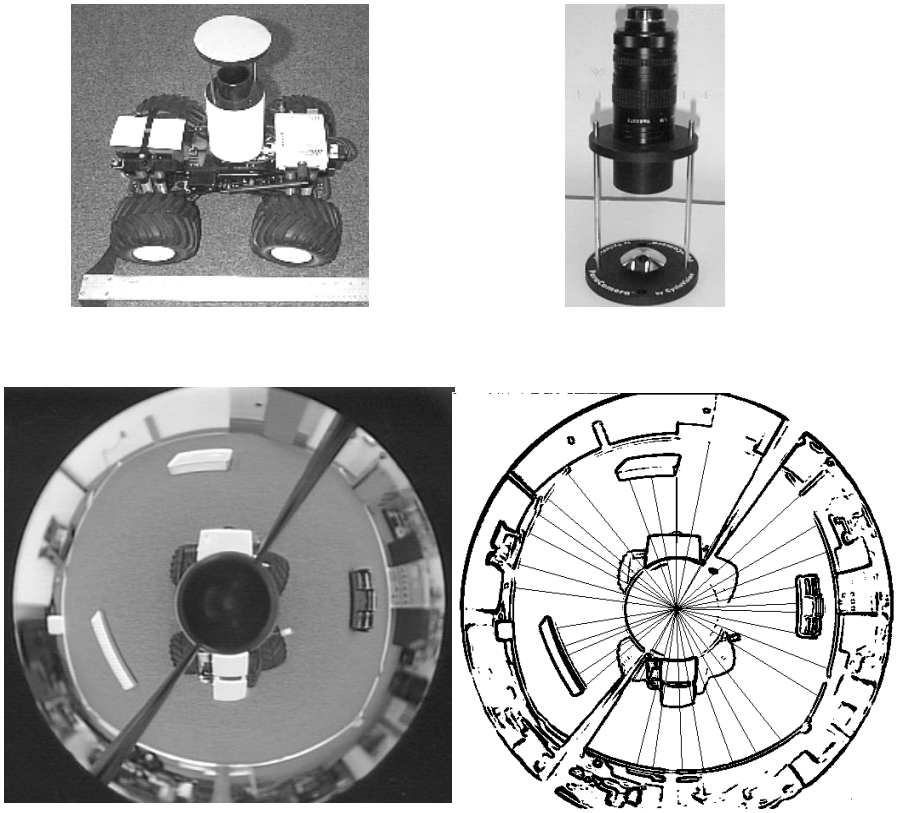

Fig. 7. The mobile robot platform with Omnicam (top), typical image and the ensuing range map (bottom).

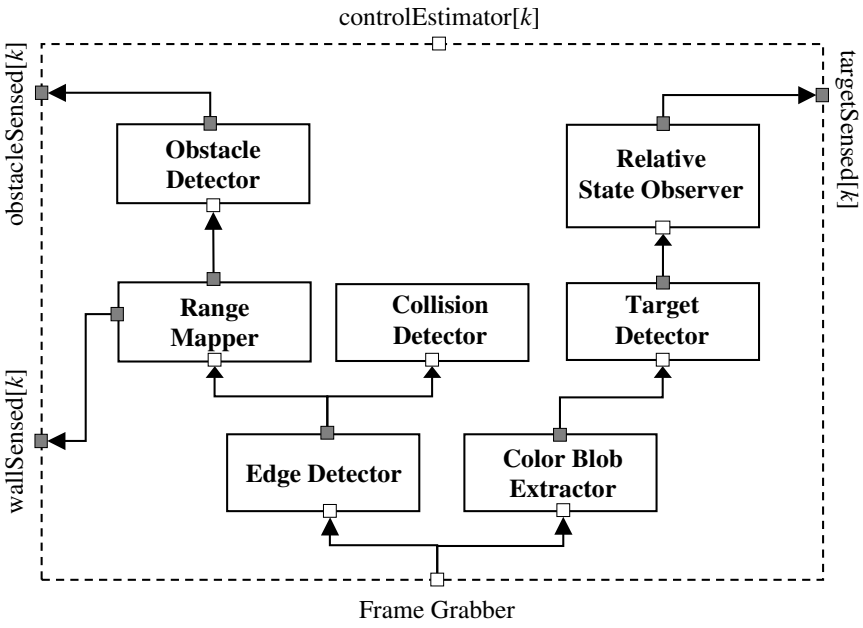

Fig. 8. Description of the *Estimator* agent.

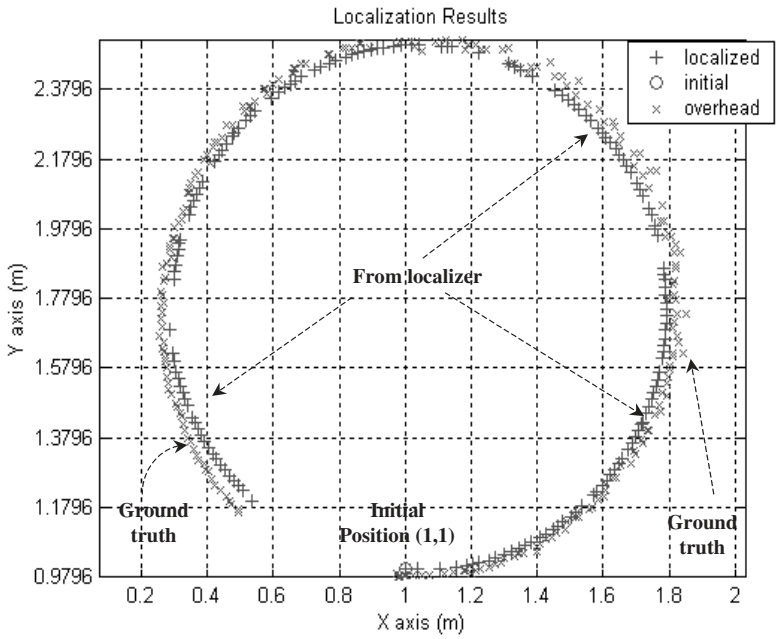

Fig. 9. Localization results for a circular trajectory.

4.2.3. Velocity Estimator

The *leader-following* control object described in the next section requires reliable estimation of the linear velocity and angular velocity of a leader mobile robot. The velocity *estimator* algorithm is also based on an EKF. It uses the *blobExtractor* sensor to determine the range ρ and the bearing β of the observed leader. In addition, the filter requires a sensor model, and a model of the dynamics of the leader and follower robots.

The EKF is based on a kinematic model of the nonholonomic mobile robot

$$\begin{aligned} \dot{x} &= u_1 \cos\theta, \quad \dot{y} = u_1 \sin\theta, \\ \dot{\theta} &= \frac{u_1}{l} \tan\phi, \quad \dot{\phi} = \lambda(u_2 - \phi), \end{aligned} \tag{2}$$

where l is the body length, u_2 is the steering command, $|\phi| < 70^\circ$ is the steering angle, and $\lambda \approx 4s^{-1}$ is a parameter that depends on the steering servo time constant and wheel-ground friction. The control vector is given by $\boldsymbol{u} = [u_1 \quad u_2]^{\mathrm{T}}$ where u_1 is the robot's forward velocity and u_2 is the steering command. More details of the EKF are provided in Das et al. (2001).

4.2.4. Mapper

When communication is enabled between the robots, centralized controllers and estimators are possible. Figure 10 shows the results of a cooperating mapper enabled by sharing information between, and coordinating, three robots. The mapper requires two robots to stay fixed at any instant, while a third robot, called the mapper, explores unknown areas. In the figure, three robots develop a map of a $4 \times 4\text{m}^2$ test area with global map updates at 3–5 Hz.

4.3. Controllers

4.3.1. Obstacle Avoidance and Wall Following

The wall following works by using inputs from two sensors: a *wall detector* and an *obstacle detector*. Both take as input the image from an *edge detector*, and use range map data to find the relative position of the wall/obstacle. The wall detector has a 40° field of view from 160°–200° with respect to the robot frame, where 90° reflects the forward direction of the robot. A line is fit to these points using a RANSAC (random sampled consensus) algorithm (Hartley and Zisserman 2000), which gives us a line fit robust to outliers. From this we are able to extract the relative position and orientation of the robot to the wall. We use input–output feedback linearization techniques to design a proportional–derivative (PD) controller to regulate the distance of the vehicle to the wall, Figure 11 (top). Wall following can be considered as a particular case of path following (De Luca, Oriolo, and Samson 1998). Thus, the kinematics, in terms of the path variables, arc length s and orientation θ_t, becomes

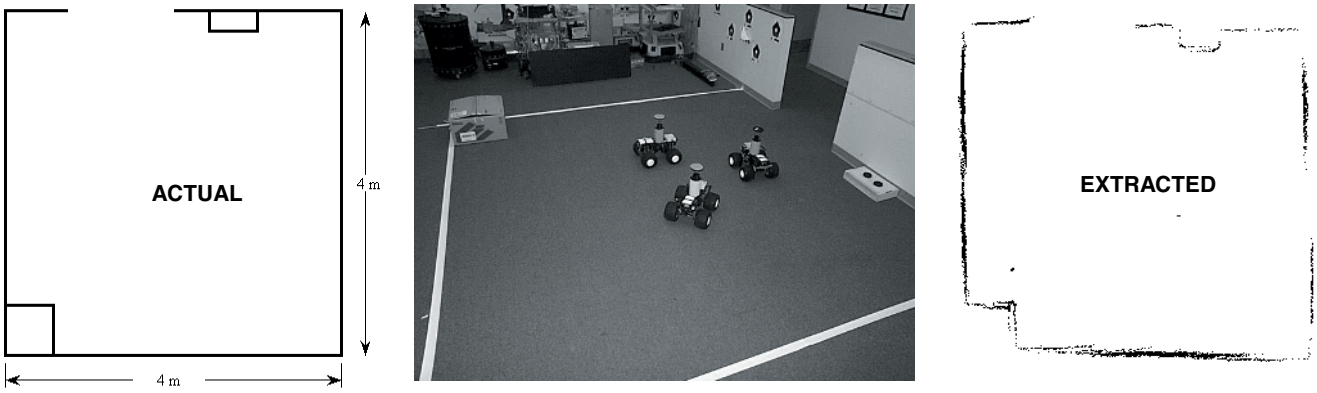

Fig. 10. Cooperative mapping with three robots, in an environment with real and simulated walls.

$$\dot{s} = v_1 \cos\theta_p, \quad \dot{d} = v_1 \sin\theta_p, \quad \dot{\theta}_p = \frac{v_1}{l}\tan\phi, \quad \dot{\phi} = v_2. \tag{3}$$

In this case $\theta_t = \frac{\pi}{2}$ and $\theta_p = \theta - \theta_t$. Assuming the robot is to follow the wall with a piecewise constant velocity $v_1(t)$, the controller is given by

$$u = \tan^{-1}\left[\frac{l}{v_1^2 \cos\theta_p}(k_p(d_0 - d) - k_v v_1 \sin\theta_p)\right], \tag{4}$$

where $u(t)$ is the steering command, $v_1(t)$ is the linear velocity, and k_p and k_v are positive design controller gains. A critically damped behavior is obtained by setting $k_v = 2\sqrt{k_p}$.

The obstacle detector picks up objects in its 80° forward-staring field of view. Since the position and orientation relative to the wall are known, the detector is able to discriminate which *obstacles* are actually the wall, and which are truly obstacles that must be avoided. Mode switching between wall following and obstacle avoidance is accomplished by giving priority to the latter. Experimental results are depicted in Figure 11 (axes units are inches).

5. Control of Groups of Robots

Problems in formation control of multiple vehicles that have been investigated include assignment of feasible formations (Tabuada, Pappas, and Lima 2001), getting into formation (Chen and Luh 1994; Beni and Liang 1996), and maintenance of formation shape (Yamaguchi and Burdick 1998). Approaches to modeling and solving these problems have been diverse, ranging from paradigms based on combining reactive behaviors (Balch and Arkin 1998; Burridge, Rizzi, and Koditschek 1999) to those based on leader-following graphs (Desai, Ostrowski, and Kumar 2001) and virtual structures (Tan and Lewis 1997; Lawton, Young, and Beard 2000).

We consider a team of n nonholonomic mobile robots that are required to follow a prescribed trajectory while maintaining a desired formation. The desired formation may change based on environmental conditions or higher-level commands. In this paper, we assume that the robots are velocity controlled platforms and have two independent inputs v_i and ω_i. The control laws are based on I/O feedback linearization. This means we are able to regulate two outputs. Moreover, we assume that the robots are assigned labels from 1 through n which restrict the choice of control laws. Robot 1 is the leader of the group. It is assumed that R_1 *knows* where to go. We do not consider explicitly the trajectory planning for the leader, but instead focus on the algorithms and software required for the group to follow the leader. Follower robots have no information about the leader's trajectory. Each *follower* robot is able to estimate the state of its leader by using extended Kalman filter techniques (Das et al. 2001). Thus, a follower can maintain a prescribed separation and bearing from its adjacent neighbors. Following our previous work (Desai, Ostrowski, and Kumar 2001), we consider two controllers that enable this behavior in the next two subsections: (a) the *Separation–Bearing Controller (SBC)*; and (b) the *Separation–Separation Controller (SSC)*.

5.1. Separation–Bearing Control

In this mode of control, the desired separations l_{ij}^d and bearings ψ_{ij}^d define the desired shape of the formation locally as shown in Figure 12 (top). The kinematics of the nonholonomic i-robot are given by

$$\dot{x}_i = v_i \cos\theta_i, \quad \dot{y}_i = v_i \sin\theta_i, \quad \dot{\theta}_i = \omega_i, \tag{5}$$

where $x_i \equiv (x_i, y_i, \theta_i) \in SE(2)$. The control velocities for the follower are given by

$$v_j = s_{ij}\cos\gamma_{ij} - l_{ij}\sin\gamma_{ij}(b_{ij} + \omega_i) + v_i\cos(\theta_i - \theta_j), \tag{6}$$

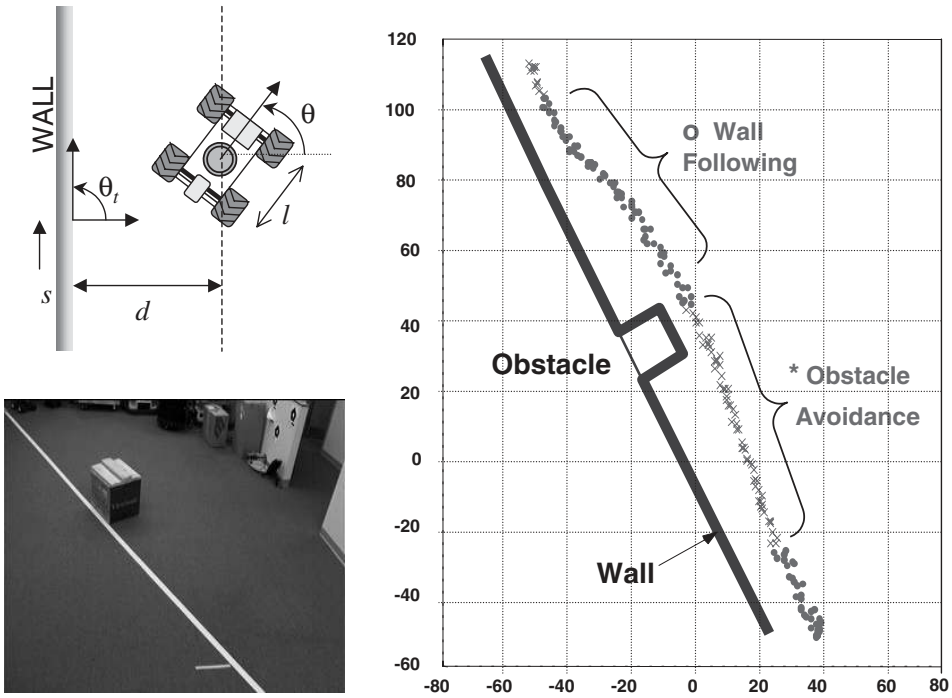

Fig. 11. The wall following, sample wall following configuration, and corresponding mode versus position results (see also Extension 3).

$$\omega_j = \frac{1}{d_j}[s_{ij}\sin\gamma_{ij} + l_{ij}\cos\gamma_{ij}(b_{ij} + \omega_i) + v_i\sin(\theta_i - \theta_j)], \tag{7}$$

where

$$\gamma_{ij} = \theta_i + \psi_{ij} - \theta_j, \tag{8}$$

$$s_{ij} = k_1(l_{ij}^d - l_{ij}), \tag{9}$$

$$b_{ij} = k_2(\psi_{ij}^d - \psi_{ij}), \qquad k_1, k_2 > 0. \tag{10}$$

The closed-loop linearized system becomes

$$\dot{l}_{ij} = k_1(l_{ij}^d - l_{ij}), \quad \dot{\psi}_{ij} = k_2(\psi_{ij}^d - \psi_{ij}), \quad \dot{\theta}_j = \omega_j. \tag{11}$$

In the following theorem, we provide a stability result for the *SBC*.

THEOREM 1. Assume that the reference trajectory $g(t)$ is smooth, the reference linear velocity is large enough and bounded, i.e., $v_i > V_{min} > 0$, the reference angular velocity is small enough, i.e., $\|\omega_i\| < W_{max}$ and the initial relative orientation is bounded, i.e., $\|\theta_i - \theta_j\| < \varepsilon_\theta < \pi$. If the control velocities (6)–(7) are applied to R_j, then system (11) is stable and the output system error of the linearized system converges to zero exponentially.

While the two output variables in eq. (11) converge to the desired values arbitrarily fast (depending on k_1 and k_2), the behavior of the follower's internal dynamics, θ_j, depends on the controlled angular velocity ω_j. In our analysis we have considered the internal dynamics required for a complete study of the stability of the system. Let the orientation error be expressed as

$$\dot{e}_\theta = \omega_i - \omega_j. \tag{12}$$

After incorporating the angular velocity for the follower (7), we obtain

$$\dot{e}_\theta = -\frac{v_i}{d_j}\sin e_\theta + \eta(\boldsymbol{w}, e_\theta), \tag{13}$$

where $\boldsymbol{w}$ depends on the output system error and reference angular velocity ω_i. $\eta(\cdot)$ is a nonvanishing perturbation for the nominal system (eq. (13) with $\eta(\cdot) = 0$), which is itself (locally) exponentially stable. By using the stability of perturbed systems (Khalil 2002), it can be shown that system (13) is stable, and thus the stability result in Theorem 1 follows.

Figure 12 shows a view of a leader-following experiment. Shown superimposed on the ground plane are actual data points collected from an overhead camera installed in our lab for ground truth purposes.

Figure 13 depicts the estimated linear and angular velocity of the leader robot, and the measured separation and bearing. We choose $l_d = 0.6$ m and $\psi_d = 180°$. The robustness of the system is verified when we manually hold the follower for a few seconds at $t \approx 65$ s.

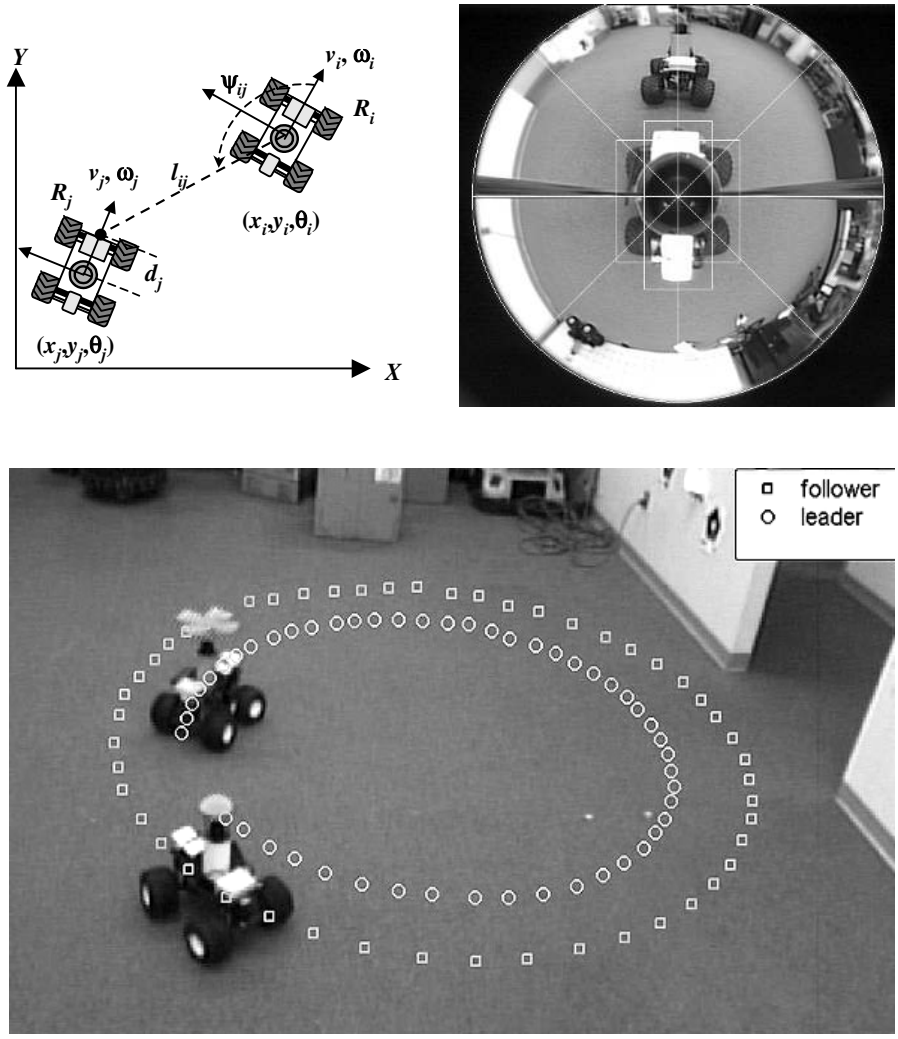

Fig. 12. The *SBC* and formation control experimental setup (see also Extension 1).

5.2. Separation–Separation Control

In this mode of control, robot R_k follows two leaders R_i and R_j with desired separations l_{ik}^d and l_{jk}^d, respectively. This controller can be used to define a triangle formation provided that R_j follows R_i with the appropriate separation and bearing (see Figure 14).

In this case the control velocities for the follower robot become

$$v_k = \frac{s_{ik}\sin\gamma_{jk} - s_{jk}\sin\gamma_{ik} + v_i\cos\psi_{ik}\sin\gamma_{jk} - v_j\cos\psi_{jk}\sin\gamma_{ik}}{\sin(\gamma_{jk}-\gamma_{ik})}, \tag{14}$$

$$\omega_k = \frac{-s_{ik}\cos\gamma_{jk} + s_{jk}\cos\gamma_{ik} - v_i\cos\psi_{ik}\cos\gamma_{jk} + v_j\cos\psi_{jk}\cos\gamma_{ik}}{d_k\sin(\gamma_{jk}-\gamma_{ik})}. \tag{15}$$

The closed-loop linearized system is

$$\dot{l}_{ik} = k_1(l_{ik}^d - l_{ik}), \quad \dot{l}_{jk} = k_1(l_{jk}^d - l_{jk}), \quad \dot{\theta}_k = \omega_k. \tag{16}$$

In the following theorem, we provide a stability result for the *SSC*. Details are described in Fierro et al. (2002).

THEOREM 2. Assume that the reference linear velocity along the trajectory $g(t) \in SE(2)$ is lower bounded, i.e., $v_i > V_{\min} > 0$, the reference angular velocity is also bounded, i.e., $\|\omega_i\| < W_{max}$, the relative velocity $\delta_v \equiv v_i - v_j$ and orientation $\delta_\theta \equiv \theta_i - \theta_j$ are bounded by small positive numbers ε_1, ε_2, and the initial relative orientation $\|\theta_i(t_0) - \theta_k(t_0)\| < \pi$. If the control velocities (14)–(15) are applied to R_k, then system (16) is stable and the output system error of the linearized system converges to zero exponentially.

Figure 15 depicts ground-truth data for triangular formation experiments. The desired formation was an isosceles triangle where both followers maintained a distance of 1.0 m

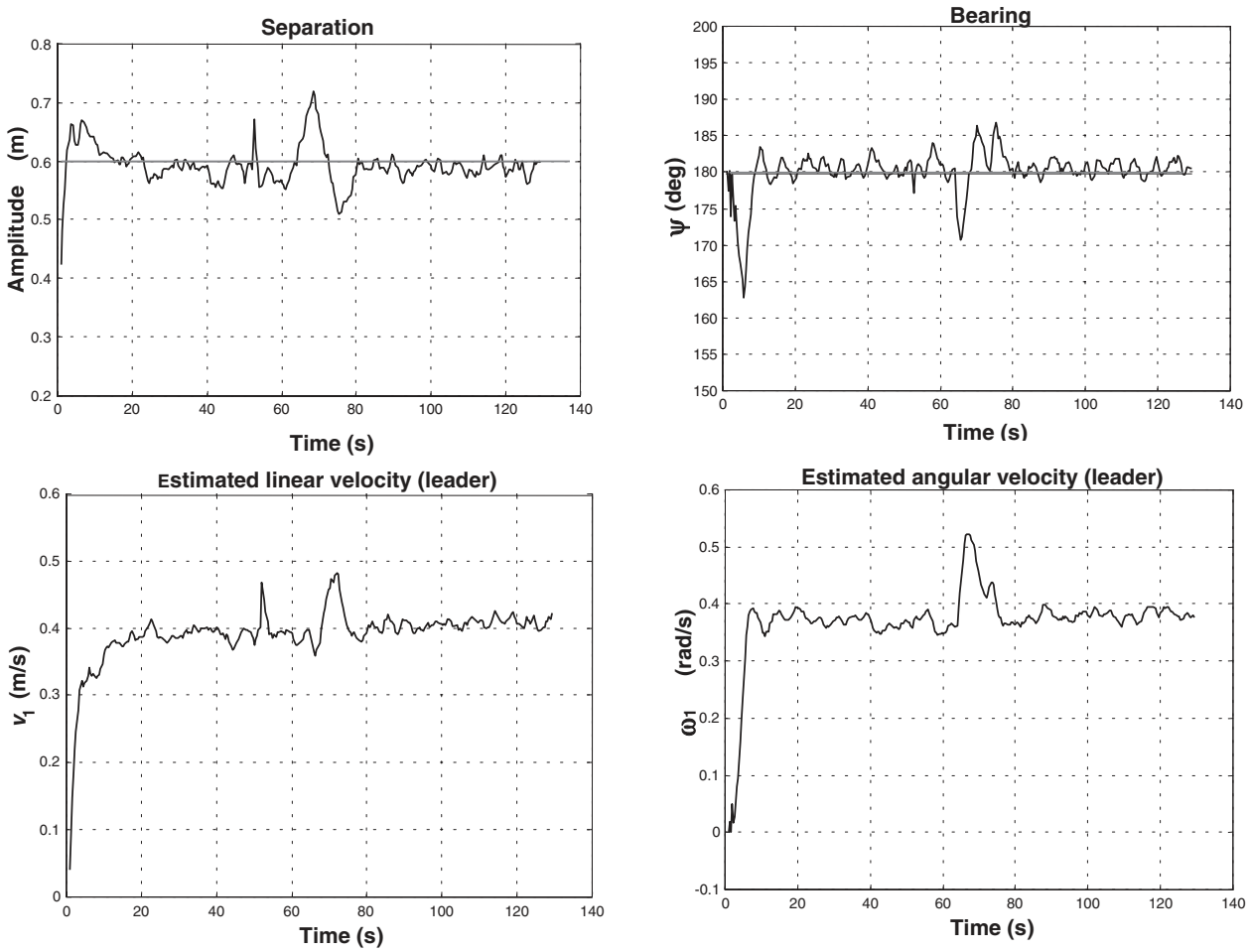

Fig. 13. Leader-following experimental results.

from the lead robot. The results are for the most part satisfactory, with mean separation errors of 3.2% and 5.5% for the two followers.

5.2.1. Distributed Manipulation

The ability to maintain a prescribed formation allows the robots to manipulate objects and transport them to a desired location. Experiments were conducted using a box as the object to be manipulated. In Figure 16, the initial team configuration is centered around the box, with the goal to flow the formation along a trajectory generated by the leader. By choosing a constraining formation geometry, the box is kept in contact with all three robots during the formation flow. Several snapshots from a sample run are shown in Figure 16.

Despite the control strategy not accounting for changes in the object pose, the formation was typically successful in its manipulation task over the tested trajectories. These experiments, while not an exhaustive investigation of distributed manipulation, demonstrate the potential for a vision-based formation control application.

5.3. Discussion

We have discussed two types of controllers for the *Leader-Following* mode for controlling and coordinating a team of mobile robots. These are not the only controllers possible. In Spletzer et al. (2001), we describe controllers that are particularly useful for cooperative localization and manipulation, and we exploit the characteristics of the omnidirectional cameras. However, as shown in Fierro et al. (2001b, 2002), these controllers can be used to control a team of n robots in arbitrarily complicated formations. Furthermore, in Fierro et al. (2001b) a simple algorithm for assigning one of the two controllers discussed above is developed. This mode-switching algorithm, although derived from heuristics based on the

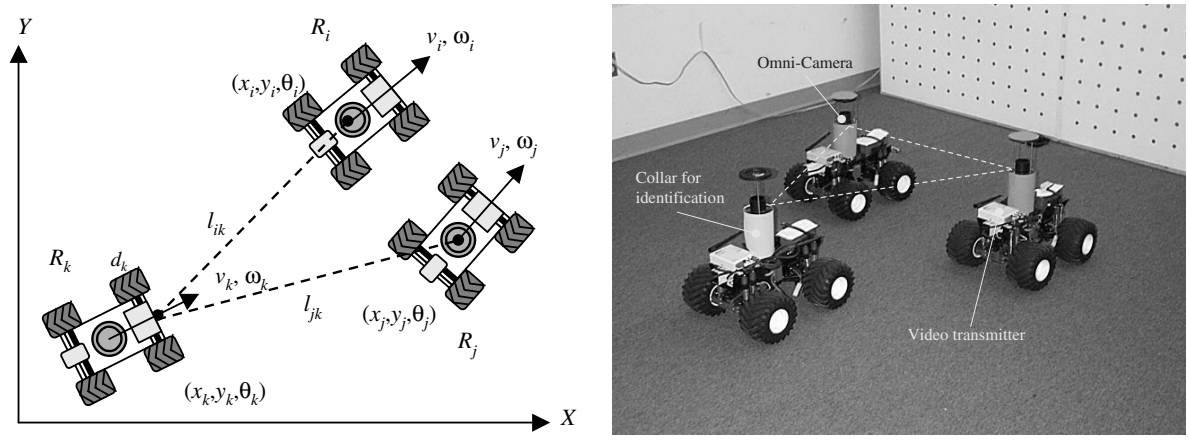

Fig. 14. The SSC and experimental setup.

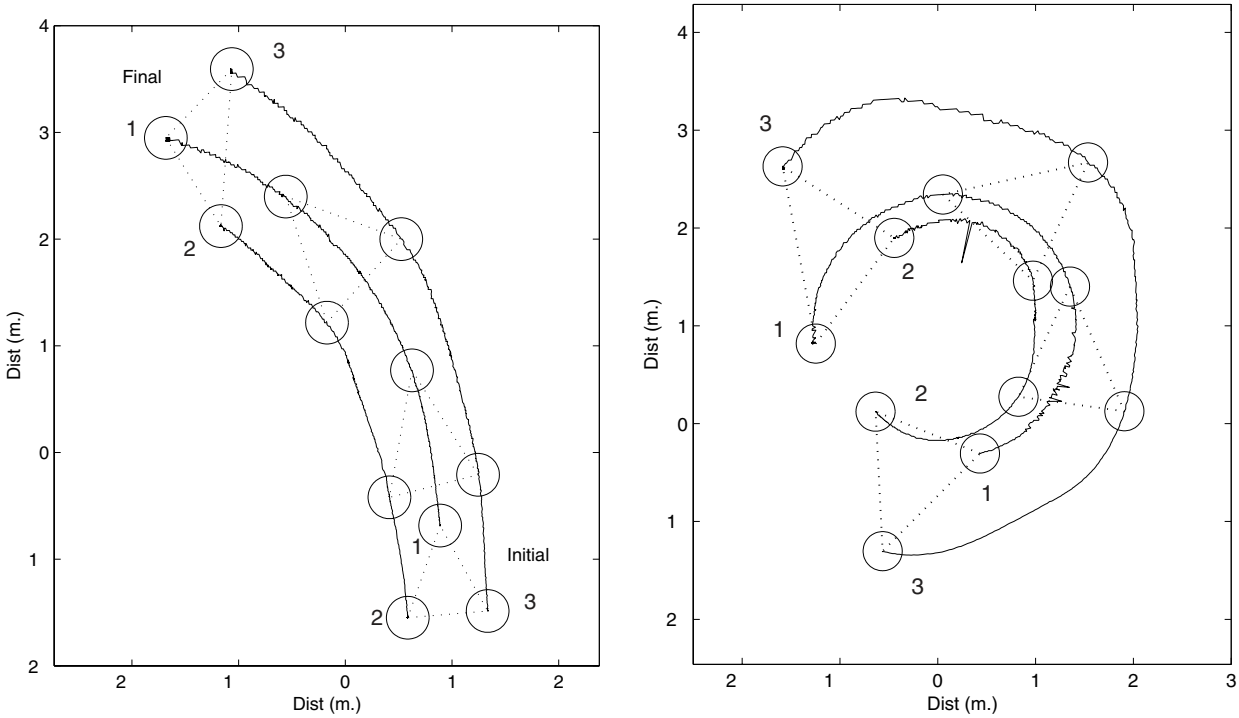

Fig. 15. Sample ground-truth data for trajectories for a triangular formation (see also Extension 5).

Fig. 16. Distributed manipulation demonstration.

sensor characteristics and the dynamics of the robot, is proven to be stable for a three-robot formation.

The assignment of controllers for an n-robot team can be formulated as an optimization problem on a directed graph, where the nodes represent the robots and each directed edge represents the assigned control policy for the associated robot pair. We call such a directed graph $\mathcal{H}$, a *control graph*. The objective for the optimization could be to maximize stability of the formation for bounded trajectories of the lead robot for a specified task. This problem is particularly complicated because the constraints on the graph must reflect the limitations of the sensor and the dynamic characteristics of the robot and are therefore continuous constraints in state space. This is a subject of ongoing work.

Finally, we assume that each robot agent can execute a finite set of modes. Thus, the tasks performed by the multi-agent system are specified as mode transition boundaries. Because it is difficult to predict exactly under what conditions switching between modes should occur, we parametrize mode boundary transitions within each robot's information space and use reinforcement reward to obtain locally optimal mode boundary locations. Given this formal specification of the control task, the autonomous agents begin to interact with the environment, collecting data. This information (e.g., samples of vehicle dynamics, obstacles, and other unknowns such as vision), and an appropriate performance index (e.g., number of mode switches, task completion time, etc.) are used within the *Boundary Localized Reinforcement Learning* (BLRL) framework to suggest an updated set of the mode boundaries. We point the reader to a description of the BLRL to obtain locally optimal mode transition boundary locations (Grudic and Ungar 2000a, 2000b).

6. Concluding Remarks

We have presented a formal architecture and high-level language for programming multiple cooperative robots. Our approach assumes that each robot has a finite set of behaviors or modes that it can execute, and the programming language is used to formally specify a set of conditions under which mode transitions take place. We have also discussed the development of stable controllers and estimators that can be used as building blocks in the bottom-up development of an intelligent system. We have described experiments involving searching and identification of targets, two-dimensional cooperative localization of targets and robots, collaborative mapping, cooperative manipulation and transportation of objects, and formation control.

Currently we are developing a three-dimensional cooperative mapping system which will be necessary for outdoor operation. Operating outdoors, as in any unstructured environment, poses challenges related to robust operation under conditions such as non-planar terrain and variable lighting conditions. Also, we are modeling the role of communication and shared information in tasks involving multi-autonomous agents. As a future work, we are planning to carry out a *reachability* analysis for our hybrid automaton framework.

We are interested in extending the applicability of our current framework to other types of multi-agent systems including unmanned aerial vehicles (UAVs) (Fierro et al. 2001a). Finally, we are very encouraged by our new approach to Reinforcement Learning (RL). It does not suffer from the curse of dimensionality that plagues most other RL work. We are actively pursuing the application of algorithms to learning group behavior.

Appendix 1: CHARON Code of a Multi-Robot Coordination Example

Consider a two-robot team that tries to arrive at a goal position in an environment with obstacles. Additionally, one robot may take the role of leader while the follower keeps a desired distance and relative bearing under separation-bearing control (Section 5). Each robot is considered to have sensors that provide partial information (or estimate) about the environment, in this case, the positions of detected obstacles.

We use the framework and architecture presented in this paper to implement this multi-robot coordination example. The CHARON code is given below.

```
/*****************
 * Multirobots
 * Version: V2.11
 *****************/
extern real Math.cos(real);
extern real Math.sin(real);
extern real Math.sqrt(real);
extern real Math.atan2(real, real);

macro single 1
macro lead 2
macro follow 3
macro pi 3.14
macro SBC 1  // Separation-Bearing Control

agent Multirobots() {
  private discrete int role1, role2,
    formation;
  private discrete bool wallDetected,
    obstacleDetected;

  agent coordination = Coordination()
              [r1c, r2c, frc, wdc, odc,
               x1, y1c, theta1c, x2c,y2c,
                 theta2c :=
               role1, role2, formation,
         wallDetected, obstacleDetected,
               x1, y1, theta1, x2, y2, theta2]
  agent robotG = RobotGroup()
              [r1rg, r2rg, frrg, wdrg, odrg,
               x1rg, y1rg, theta1rg, x2rg,
                 y2rg, theta2rg :=
               role1, role2, formation,
                 wallDetected, obstacleDetected,
                 x1, y1, theta1, x2, y2, theta2]
}

agent Coordination() {
  read analog real x1c, y1c, theta1c, x2c, y2c,
    theta2c;
  write discrete int r1c, r2c, frc;
  write discrete bool wdc, odc;

  mode top = CoordinationTop();
  init {r1c = lead; r2c = follow; frc = SBC;
        wdc = false; odc = false}
}

agent RobotGroup() {
  read discrete int r1rg, r2rg, frrg;
  read discrete bool wdrg, odrg;
  write analog real x1rg, y1rg, theta1rg, x2rg,
    y2rg, theta2rg;

  agent robot1 = Robot(2.0, 2.0, 0.0,
                       0.5, 0.0, 0.6, 3.14, 0.1)
              [role, frr, wdr, odr, x, y, v,
                theta, omega :=
           r1rg, frrg, wdrg, odrg, xL, yL, vL,
             thetaL, omegaL]
  agent robot2 = Robot(0.0, 0.0, 0.0,
                       0.0, 0.0, 0.6, 3.14, 0.1)
              [role, frr, wdr, odr, xI, yI, vI,
                thetaI, omegaI :=
           r2rg, frrg, wdrg, odrg, xL, yL, yL,
              thetaL, omegaL]
}

agent Robot(real initX, real initY, real init
Theta,
            real initV, real initOmega,
              real ld, real psid,
            real d) {
  read discrete int role, frr;
  read discrete bool wdr, odr;
  read analog real  xI, yI, vI, thetaI, omegaI;
  write analog real x, y, v, theta, omega;

  mode top = ControllerTop
              (initX, initY, initTheta,
               initV, initOmega, ld, psid, d)
}

mode ControllerTop (real initX, real initY,
  real initTheta,
            real initV, real initOmega,
              real ld, real psid,
            real d) {
  read discrete int role;
  write analog real x, y, v, theta, omega;
  private analog real t;

  mode lead = LeaderMode()
```

```
  mode follow = FollowerMode(ld, psid, d, 1.0,
    0.5, 1.0, 3.14/10.0)
                //k1, k2, deltaL, deltaPsi

  trans from default to lead when role==lead
        do {x=initX; y=initY; v=initV;
            theta=initTheta; omega=initOmega;
              t=0 }
  trans from default to follow when role=
     =follow
        do {x=initX; y=initY; v=initV;
            theta=initTheta; omega=initOmega;
              t=0 }
}

mode LeaderMode(){
  entry ptN;
  exit ptX;

  read discrete bool wallDetected,
    obstacleDetected;
  write analog real x, y, v, theta, omega;

  diff dTimer {d(t) == 1.0}

  mode goToGoal = GoToGoalMode()
  mode wallFollowing = WallFollowingMode()
  mode obstacleAvoidance =
    ObstacleAvoidanceMode()

  trans from default to goToGoal when true
    do {}
  trans from goToGoal.ptX to wallFollowing.
    ptN
        when {wallDetected == true} do {}
        to obstacleAvoidance.ptN
        when {wallDetected == false &&
          obstacleDetected == true}
        do {}
  trans from wallFollowing.ptX to
    goToGoal.ptN
        when {wallDetected == false} do {}
        to obstacleAvoidance.ptN
        when {obstacleDetected == true} do {}
  trans from obstacleAvoidance.ptX to
    goToGoal.ptN
        when {wallDetected == false &&
          obstacleDetected == false}
        do {}
        to  wallFollowing.ptN
        when {wallDetected == true &&
          obstacleDetected == false}
        do {}
}

mode GoToGoalMode(){
  entry ptN;
  exit ptX;
  read discrete bool wallDetected,
    obstacleDetected;
  read analog real t;
  write analog real x, y, v, theta, omega;

  inv invGoToGoal {wallDetected == false &&
    obstacleDetected == false}
  diff dLeadXYT {d(x) == v*Math.cos(theta);
                 d(y) == v*Math.sin(theta);
                 d(theta) == omega}
  diff dVO {d(v) == 0; d(omega) ==
    0.1*Math.sin(0.1*t)}
}

mode followerMode(real ld, real psid, real d,
real k1, real k2,
                real deltaL, real deltaPsi){
  entry ptN;
  exit ptX;

  write analog real x, y, v, theta, omega;
  read analog real xI, yI, vI, thetaI, omegaI;
  private analog real l, psi;

  inv invFollow {Math.abs(l-ld) <= deltaL
                 and Math.abs(psi-psid) <=
                   deltaPsi}
  diff dFollowV {d(v) == k1*(ld-l)*Math.cos
    (thetaI+psi-theta)
          -l*Math.sin((thetaI+psi-theta)*
            (k2*(psid-psi)+omegaI))
          +vI*Math.cos(thetaI-theta)}
  diff dFollowOmega {d(omega) == 1/d*(k1*
    (ld-l)*Math.sin(thetaI+psi-theta)
          +l*Math.cos((thetaI+psi-theta)*(k2*
            (psid-psi)+omegaI))
          +vI*Math.sin(thetaI-theta))}
  diff dFollowO {d(theta) == omega }
  diff dFollowXY {d(x) == v*Math.cos(theta);
                  d(y) == v*Math.sin(theta) }
  alge aLP {l == Math.sqrt((x-xI)(x-xI)+(y-yI)
    (y-yI));
           psi == pi-thetaI-Math.atan2(y-yI,
             xI-x))}
}
```

Appendix 2: Index to Multimedia Extensions

The multimedia extension page is found at http://www.ijrr.org.

Table of Multimedia Extensions

Extension	Type	Description
1	Video	Experimental platform and obstacle avoidance experiment

(Continued on next page)

2	Video	In this experiment we restrict the robot to two modes: (a) target acquisition and (b) target tracking
3	Video	Switching between modes: obstacle avoidance and line (simulated wall) following
4	Video	Leader-following. The follower uses an extended Kalman filer (EKF) for estimating its leader's velocities.
5	Video	A 3-robot system in triangular formation
6	Video	Collaborative manipulation

Acknowledgments

This research was supported in part by DARPA ITO MARS 130-1303-4-534328-xxxx-2000-0000, DARPA ITO MoBIES F33615-00-C-1707, and NSF grant CISE/CDS-9703220. We thank Dan Walker for his work on the hardware of our experimental mobile platform. We also thank the anonymous reviewers for their constructive comments to improve this paper.

References

Alur, R., Grosu, R., Hur, Y., Kumar, V., and Lee, I. 2000a. Modular specifications of hybrid systems in CHARON. In Lynch, N. A. and Krogh, B. H., eds., *Hybrid Systems: Computation and Control, LNCS 1790*, pp. 6–19. Berlin: Springer.

Alur, R., Henzinger, T., Lafferriere, G., and Pappas, G. 2000b. Discrete abstractions of hybrid systems. *Proc. IEEE*, 88(7):971–984.

Arkin, R. 1998. *Behavior-Based Robotics*. MIT Press.

Baker, S., and Nayar, S. 1998. A theory of catadioptric image formation. In *International Conference on Computer Vision*, pp. 35–42, Bombay, India.

Balch, T., and Arkin, R. 1998. Behavior-based formation control for multi-robotic teams. *IEEE Transactions on Robotics and Automation*, 14(6):926–934.

Belta, C., and Kumar, V. 2001. Motion generation for formations of robots: A geometric approach. In *Proc. IEEE Int. Conf. on Robotics and Automation*, pp. 1245–1250, Seoul, Korea.

Beni, G., and Liang, P. 1996. Pattern reconfiguration in swarms—convergence of a distributed asynchronous and bounded iterative algorithm. *IEEE Transactions on Robotics and Automation*, 12(3):485–490.

Brooks, R. 1986. A robust layered control system for a mobile robot. *IEEE Journal on Robotics and Automation*, 2(1):14–23.

Burgard, W., Moors, M., Fox, D., Simmons, R., and Thrun, S. 2000. Collaborative multi-robot exploration. In *Proc. IEEE Int. Conf. on Robotics and Automation*, pp. 476–481, San Francisco, CA.

Burridge, R., Rizzi, A., and Koditschek, D. 1999. Sequential composition of dynamically dexterous robot behaviors. *International Journal of Robotics Research*, 18(6):534–555.

Chen, Q., and Luh, J. Y. S. 1994. Coordination and control of a group of small mobile robots. In *Proc. IEEE Int. Conf. on Robotics and Automation*, Vol. 3, pp. 2315–2320.

Das, A. K., Fierro, R., Kumar, V., Ostrowski, J. P., Spletzer, J., and Taylor, C. J. 2002. A vision-based formation control framework. *IEEE Transactions on Robotics and Automation*, 18(5):813–825.

Das, A. K., Fierro, R., Kumar, V., Southall, B., Spletzer, J., and Taylor, C. 2001. Real-time vision-based control of a nonholonomic mobile robot. In *Proc. IEEE Int. Conf. on Robotics and Automation*, pp. 1714–1719, Seoul, Korea.

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

Desai, J., Kumar, V., and Ostrowski, J. P. 1999. Control of changes in formation for a team of mobile robots. In *Proc. IEEE Int. Conf. on Robotics and Automation*, pp. 1556–1561, Detroit, Michigan.

Desai, J. P., Ostrowski, J. P., and Kumar, V. 2001. Modeling and control of formations of nonholonomic mobile robots. *IEEE Transactions on Robotics and Automation*, 17(6):905–908.

Donald, B., Gariepy, L., and Rus, D. 2000. Distributed manipulation of multiple objects using ropes. In *Proc. IEEE Int. Conf. on Robotics and Automation*, pp. 450–457.

Feddema, J., and Schoenwald, D. 2001. Decentralized control of cooperative robotic vehicles. In *Proc. SPIE*, Vol. 4364, *Aerosense*, Orlando, FL.

Fierro, R., Belta, C., Desai, J., and Kumar, V. 2001a. On controlling aircraft formations. In *Proc. IEEE Conf. on Decision and Control*, pp. 1065–1070, Orlando, FL.

Fierro, R., Das, A., Kumar, V., and Ostrowski, J. P. 2001b. Hybrid control of formations of robots. In *Proc. IEEE Int. Conf. on Robotics and Automation*, pp. 157–162, Seoul, Korea.

Fierro, R., and Lewis, F. L. 1997. A framework for hybrid control design. *IEEE Trans. Syst. Man Cybern.*, 27-A(6):765–773.

Fierro, R., Song, P., Das, A. K., and Kumar, V. 2002. Cooperative control of robot formations. In Murphey, R. and Pardalos, P., eds., *Cooperative Control and Optimization*, Vol. 66 of *Applied Optimization*, chapter 5, pp. 73–93. Dordrecht: Kluwer Academic.

Grudic, G. Z., and Ungar, L. H. 2000a. Localizing policy gradient estimates to action transitions. In *Proc. 17th Int. Conf. on Machine Learning*, Vol. 17, pp. 343–350. San Mateo, CA: Morgan Kaufmann.

Grudic, G. Z., and Ungar, L. H. 2000b. Localizing search in reinforcement learning. In *Proc. 17th National Conf. on Artificial Intelligence*, Vol. 17, pp. 590–595. Menlo Park, CA: AAAI/Cambridge, MA: MIT Press.

Hartley, R., and Zisserman, A. 2000. *Multiple View Geometry in Computer Vision*. Cambridge: Cambridge University Press.

Jennings, J. S., Whelan, G., and Evans, W. F. 1997. Cooperative search and rescue with a team of mobile robots. In *Proc. IEEE Int. Conf. on Advanced Robotics (ICAR)*, pp. 193–200, Monterey, CA.

Khalil, H. K. 2002. *Nonlinear Systems*, 3rd edition. Upper Saddle River, NJ: Prentice Hall.

Khatib, O., Yokoi, K., Chang, K., Ruspini, D., Holmberg, R., and Casal, A. 1996. Vehicle/arm coordination and mobile manipulator decentralized cooperation. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pp. 546–553.

Klavins, E., and Koditschek, D. 2000. A formalism for the composition of concurrent robot behaviors. In *Proc. IEEE Int. Conf. on Robotics and Automation*, Vol. 4, pp. 3395–3402, San Francisco, CA.

Kosuge, K., Hirata, Y., Asama, H., Kaetsu, H., and Kawabata, K. 1999. Motion control of multiple autonomous mobile robots handling a large object in coordination. In *Proc. IEEE Int. Conf. on Robotics and Automation*, pp. 2666–2673, Detroit, MI.

Lawton, J., Young, B., and Beard, R. 2000. A decentralized approach to elementary formation maneuvers. In *Proc. IEEE Int. Conf. on Robotics and Automation*, pp. 2728–2733, San Francisco, CA.

Matarić, M. 1995. Issues and approaches in the design of collective autonomous agents. *Robotics and Autonomous Systems*, 16(2-4):321–331.

Matarić, M., Nilsson, M., and Simsarian, K. 1995. Cooperative multi-robot box pushing. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pp. 556–561, Pittsburgh, PA.

Morrow, J., and Khosla, P. 1997. Manipulation task primitives for composing robot skills. In *Proc. IEEE Int. Conf. on Robotics and Automation*, Vol. 4, pp. 3354–3359.

Parker, L. 1998. ALLIANCE: An architecture for fault tolerant multi-robot cooperation. *IEEE Transactions on Robotics and Automation*, 14(2):220–240.

Parker, L. E. 2000. Current state of the art in distributed autonomous mobile robotics. In Parker, L. E., Bekey, G., and Barhen, J., eds., *Distributed Autonomous Robotic Systems*, Vol. 4, pp. 3–12. Tokyo: Springer.

Roumeliotis, S., and Bekey, G. 2000. Collective localization: A distributed kalman filter approach to localization of groups of mobile robots. In *Proc. IEEE Int. Conf. on Robotics and Automation*, pp. 2958–2965, San Francisco, CA.

Rus, D., Donald, B., and Jennings, J. 1995. Moving furniture with teams of autonomous robots. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pp. 235–242, Pittsburgh, PA.

Schreckenghost, D., Bonasso, P., Kortenkamp, D., and Ryan, D. 1998. Three tier architecture for controlling space life support systems. In *Proc. IEEE Int. Joint Symposia on Intelligence and Systems*, pp. 195–201, Rockville, MD.

Simmons, R., Singh, S., Hershberger, D., Ramos, J., and Smith, T. 2000. Coordination of heterogeneous robots for large-scale assembly. In *Proc. ISER00, 7th Int. Symposium on Experimental Robotics*, pp. 311–320, Honolulu, Hawaii.

Spletzer, J., Das, A., Fierro, R., Taylor, C. J., Kumar, V., and Ostrowski, J. P. 2001. Cooperative localization and control for multi-robot manipulation. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pp. 631–636, Maui, Hawaii.

Stilwell, D., and Bay, J. 1993. Toward the development of a material transport system using swarms of ant-like robots. In *Proc. IEEE Int. Conf. on Robotics and Automation*, pp. 766–771, Atlanta, GA.

Stroupe, A., Martin, M., and Balch, T. 2001. Distributed sensor fusion for object position estimation by multi-robot systems. In *Proc. IEEE Int. Conf. on Robotics and Automation*, pp. 1092–1098, Seoul, Korea.

Sugar, T., and Kumar, V. 2000. Control and coordination of multiple mobile robots in manipulation and material handling tasks. In Corke, P. and Trevelyan, J., eds., *Experimental Robotics VI: Lecture Notes in Control and Information Sciences*, Vol. 250, pp. 15–24. Berlin: Springer-Verlag.

Tabuada, P., Pappas, G., and Lima, P. 2001. Feasible formations of multi-agent systems. In *Proc. American Control Conference*, pp. 56–61, Arlington, VA.

Tan, K. H., and Lewis, M. A. 1997. Virtual structures for high precision cooperative mobile robot control. *Autonomous Robots*, 4:387–403.

Taylor, C. J. 2002. Videoplus: a method for capturing the structure and appearance of immersive environments. *IEEE Transactions on Visualization and Computer Graphics*, 8(2):171–182.

van der Schaft, A. and Schumacher, H. 2000. *An Introduction to Hybrid Dynamical Systems*, Vol. 251 of *Lecture Notes in Control and Information Sciences*. London: Springer.

Volpe, R., Nesnas, I., Estlin, T., Mutz, D., Petras, R., and Das, H. 2001. The CLARAty architecture for robotic autonomy. In *Proc. IEEE Aerospace Conference*, Vol. 1, pp. 121–132, Big Sky, MT.

Yamaguchi, H., and Burdick, J. W. 1998. Asymptotic stabilization of multiple nonholonomic mobile robots forming groups formations. In *Proc. IEEE Int. Conf. on Robotics and Automation*, pp. 3573–3580, Leuven, Belgium.