Proceedings of the 2001 IEEE
International Conference on Robotics & Automation
Seoul, Korea - May 21-26, 2001

An Architecture for Tightly Coupled Multi-Robot Cooperation

Luiz Chaimowicz[1,3], Thomas Sugar[2], Vijay Kumar[1], and Mario F. M. Campos[3]

[1]GRASP Laboratory - University of Pennsylvania, Philadelphia, PA, USA, 19104
[2]Mechanical and Aerospace Engineering - Arizona State University, Tempe, AZ, USA, 85283
[3]DCC - Universidade Federal de Minas Gerais, Belo Horizonte, MG, Brasil, 31270-901
{chaimo, kumar}@grasp.cis.upenn.edu, thomas.sugar@asu.edu, mario@dcc.ufmg.br

Abstract

This paper proposes a new architecture for tightly coupled multi-robot coordination that is well suited to cooperative manipulation tasks. At all times, a robot is identified as a leader, while the others are designated as followers. The assignment of roles and the coordination between the robots is guaranteed by communication protocols and control algorithms. The key feature is the flexibility that allows changes in leadership and assignment of roles during the execution of a task. We describe the experimental implementation and demonstration in a cooperative transportation task, in which two and three heterogeneous robots cooperate to carry a large object in an environment containing obstacles.

1 Introduction

Cooperative robotics has been an active research field in the last few years. The use of multiple robots working in coordination to execute different types of tasks can bring several advantages over a single robot solution such as simplicity in robot design, better performance, increased fault tolerance and spatially distributed sensing and actuation.

Many robotic tasks can be executed by single robots. However, in general, multi-robot teams can accomplish these tasks using simpler and less expensive robots. Further, multi-robot teams are inherently more flexible - they can be reconfigured and adapted to perform many tasks. An important class of tasks for robot cooperation are those that *cannot* be accomplished by a single robot and require real-time coordinated control between robots for execution. In this kind of cooperation, referred to in this paper as *tightly coupled cooperation*, the robots must act in a highly coordinated fashion in order to complete the mission. This implies that the robots must have some knowledge about the state and actions of its teammates, either through sensory perception or explicit communication. Further, in many cases, each robot is critical to the task. If one of the robots is not able to perform its subtask, the task cannot be completed as specified and the robots must be re-tasked appropriately.

In this paper, an architecture for tightly coupled multi-robot coordination for cooperative manipulation tasks is proposed. Using communication protocols and distributed control, a team of heterogeneous robots can execute tasks that require tight coordination. We assume that one robot in the team is the leader, and it determines the trajectory that must be followed. However, our role assignment mechanism enables a dynamic reconfiguration of the robots. At any moment, the robot that is leading can become a follower, and any follower can take over the leadership. The dynamic reconfiguration allows the robots to react to unexpected events such as the detection of obstacles or robot failures.

In spite of the large number of works in cooperative robotics (see [3] and [12] for surveys), the specific problem of tightly coupled cooperation is still an open research field. The majority of works in this area have focused on a specific task: material handling and transportation by multiple robots. The term "strong cooperation" was introduced in [2], meaning that the robots must act in concert to achieve the goal. In contrast to previous work which stresses on complex dynamical models [8], the use of force sensing [7] and homogeneous robots [9], the present work focuses on the use of communication in conjunction with simple control algorithms, thus enabling completely heterogeneous robots, with different operational systems, driving mechanisms and sensor power, to work in coordination to solve complex tasks.

Communication, adaptability and fault tolerance are important aspects of cooperative architectures. Behavior based approaches have been used to study these aspects. In [10], two six-legged robots and a task sharing paradigm were used to show that communication can improve performance and compensate for sensory limitations. In [11] there is a description of Alliance, a behavior-based fault tolerant architecture for heterogeneous multi-robot cooperation. New cooperative approaches to the problem of material transportation by multiple robots include the manipulation of an object using ropes [5] and the cooperation with a human agent [6].

0-7803-6475-9/01/$10.00© 2001 IEEE

Our work builds on previous work [13, 15, 16] in which a decentralized control system was developed for coordinated control of mobile manipulators. In [15], the use of passive compliance (realized by an actively controlled parallel manipulator) was shown to result in a robust grasp. As shown in a companion paper [14], the basic framework allows the team of robots to follow a lead robot. The leader plans a trajectory and broadcasts it to the followers, that control their trajectory based on this plan and feedback from their position, velocity and sensors. The main contribution of this paper is the development of a software architecture that allows the reassignment of the leader and possibly the relabeling of follower robots, which allows such tightly coupled manipulation systems to be more adaptive and more robust. In this new architecture, the robots are capable of exchanging roles during the cooperation, adapting themselves better to the task requirements and dynamic events in the environment. It incorporates the flexibility and adaptability of behavior based architectures into the real-time, coordinated control framework that is important for tightly coupled cooperation tasks. The effectiveness is demonstrated in a cooperative transportation task, in which two and three heterogeneous robots work together to carry a large object in a environment containing obstacles. This work also takes the first step toward a hybrid systems framework for modeling the cooperation between robots and the specification of continuous controllers and discrete coordination protocols.

This paper is organized as follows. The next section explains the architecture in detail. In Section 3, the testbed for the application is described and in Section 4 the experimental results are shown. Section 5 gives a summary and directions for future work.

2 Architecture

A diagram of the architecture is shown in Figure 1. The assigned leader has a planner and broadcasts its estimated position and velocity (data messages) to all the followers. Each follower has its own trajectory controller that acts in order to cooperate with the leader. The planner and the trajectory controllers send set points to the low level controllers that are responsible for the actuators. All the robots have a coordination module that controls the cooperative execution of the task. This module receives information from the sensors and exchanges control messages with the other robots. It is responsible for the role assignment and for other decisions that directly affect the planners and trajectory controllers.

The followers are capable of cooperating because they know the task and the leader's intentions. If they cannot cooperate, they will be able to request

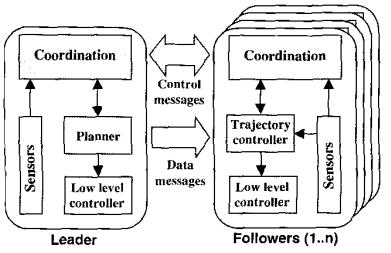

Figure 1: Architecture for tightly coupled cooperation

for a role reassignment using explicit communication.

2.1 Role Assignment

The role assignment mechanism allows the robots to exchange roles during the cooperation reconfiguring dynamically their coordination patterns. The main purpose is to adapt the robots and the cooperation to unexpected events such as obstacle detection, sensor failures, etc. It is also important to divide the leadership among the robots in such a way that, in each phase of the cooperation, the robot that is best suited in terms of sensor power, manipulation capabilities, etc., will be leading the group.

At all times, a robot is identified as a leader while the others are designated as followers. Basically, there are two methods for changing the leadership. A follower can request the leadership or a leader can resign it. In the leadership request, a follower sends a message requesting the leadership when it is not able to follow the leader's plan or when it sees a clear path to the goal. For example, if one of the followers detects an obstacle, it can request the leadership, avoid the obstacle, and then return the leadership to the previous leader. A leader can also relinquish the leadership to another robot. This can happen when the robot senses that it is unable to continue leading or when it finishes its leading turn in a task that has more than one leader. Sometimes it may be necessary to do some negotiation to decide which robot in the team will be the leader. This is discussed later in Section 2.2.

2.2 Multiple Leaders

Figure 1 describes an architecture with one leader and many followers. As described in [4], it is possible to generalize this architecture to allow multiple leaders. While the team has one designated *lead robot*, there may be many leaders. A follower can be a leader for another follower, as shown in Figure 2. These leader-follower interactions or controllers are best described by a directed, acyclic graph as in [4]. In this paper, we will only consider a small team of robots in which there is only one leader and all other robots follow the leader.

The dynamic role assignment protocol may lead to conflicts that need to be resolved. Two examples in

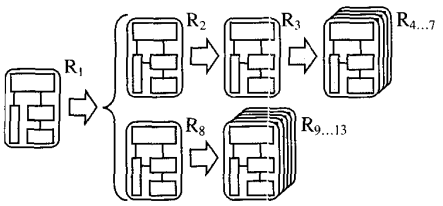

Figure 2: Example of an architecture with one team leader (R_1) and three followers that are also leaders (R_2, R_3, R_8)

the small team with one leader are: (a) a robot requests leadership but the leader does not relinquish its leadership; and (b) a robot resigns its leadership, but there are no takers. A related problem is the possibility of a chattering phenomenon where changes in leadership occur too frequently. A priority based approach is necessary to resolve such conflicts. In our present system, we detect deadlocks and, in such situations, the command is relinquished to the human operator whose authority supersedes other robots.

2.3 Communication

The robots are able to communicate exchanging messages through wireless Ethernet. They use IPX, a connectionless datagram protocol. The robots do not have to establish a connection to exchange packages and each package is treated as an individual entity, having no logical or sequential relation to another package. Thus, IPX packages are addressed and sent to their destination, but there is no guarantee or verification of successful delivery. Because of these characteristics, IPX is a fast and simple communication protocol being suitable for this kind of application.

In this architecture, all messages are broadcast using specific sockets and received by all the robots. There are two types of messages: data and control messages. Data messages are continuously broadcast by the leader and contain its estimate of its current position and intended velocity. This information can be used by the followers to control their positions or velocities in relation to the leader. Control messages are exchanged among the robots to coordinate the role assignment and other discrete control issues such as task initialization and completion. Because IPX does not provide delivery confirmation, an acknowledgment mechanism was developed. This is not necessary for data messages, that are continuously sent, but is important for control messages, that are used in the task control.

3 The Team of Robots

3.1 Experimental Setup

The architecture described in the last section is used to coordinate multiple robots carrying a box. Three heterogeneous robots with different sensing capabilities, driving mechanisms and operating systems are used in the experiments. The first robot is a TRC Labmate platform, equipped with an actively controlled compliant arm. The platform is non-holonomic, and the only on-board sensors are encoders located at the arm and at the two actuated wheels. All the programming is done using Simulink and Real Time Workshop and compiled for DOS. The second robot is a XR4000, developed by Nomadic Technologies. It has a holonomic driving system offering three degrees of freedom and is equipped with several types of sensors and a fork-lift arm. It uses the Linux system and the programming is done using C. The third robot is a Nomad Super Scout II. In the same way as the XR4000, it has a Pentium-based PC processor running Linux, but it is a non-holonomic differential drive robot. The Scout is equipped with ultrasound and contact sensors, but it does not have manipulation capabilities. All robots are equipped with wireless Ethernet boards.

3.2 Modeling

Different controllers and planners are used by each robot depending on its role in the task. The two robots carrying a box must tightly coordinate themselves to keep the balance of the box while they are moving to the desired position. When the third robot is used as a remote sensor, it must maintain a certain formation in order to detect possible obstacles.

The cooperative system can be described by its state ($\boldsymbol{X}$), that is a composition of the states of the three robots:

$$\boldsymbol{X} = [\boldsymbol{x_1}, \boldsymbol{x_2}, \boldsymbol{x_3}]^T, \quad \dot{\boldsymbol{X}} = F(\boldsymbol{X}, t).$$

The state of each robot varies as a function of its continuous state ($\boldsymbol{x}$) and the input vector ($\boldsymbol{u}$). The input vector depends on the discrete state of the robot (q), also called a mode. The input is determined by a control law that is a function of the robot's continuous state, the time and the information about the rest of the system ($\hat{\boldsymbol{z}}$)[1].

$$\begin{array}{l} \dot{\boldsymbol{x}} = f(\boldsymbol{x}, \boldsymbol{u}) \\ \boldsymbol{u} = g_q(\boldsymbol{x}, \hat{\boldsymbol{z}}, t) \end{array} \Rightarrow \dot{\boldsymbol{x}} = f_q(\boldsymbol{x}, \hat{\boldsymbol{z}}).$$

To determine the kinematic equations and inputs it is necessary to consider the characteristics of each robot in each mode. We will consider the specific task in which the robots collectively grasp and transport a designed object. The robots can be in one of the following modes: Dock, where they must coordinate themselves to grasp and pick up the object, and Transport, where they march in a coordinated fashion. Figure 3 presents these modes using behavioral hierarchy

[1] On our notation, ($\hat{\cdot}$) represents estimates of the state of the other robots.

diagrams. Diagram (a) shows the high level modes of the task. The Transport mode consists of two submodes Lead, and Follow, as shown in Diagram (b). Diagram (c) presents the submodes of the XR4000's Dock mode. In this task, the Transport mode is similar for all robots, but the Dock mode depends on the sensing and manipulation capabilities of each one.

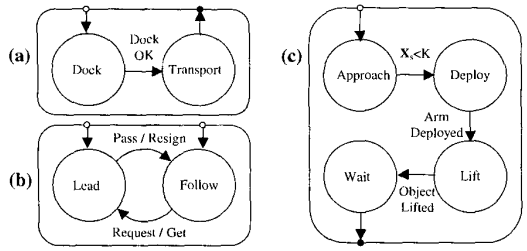

Figure 3: Behavioral hierarchy diagrams - (a) High Level Modes, (b) Transport Mode, (c) Dock Mode

Figure 4 shows a diagram of the two robots carrying an object in an environment with obstacles. Since the Labmate is non-holonomic the inputs for the low level controllers are the linear and angular velocities ($u_1 = v$, $u_2 = \omega$). Thus, the state equations become:

$$\dot{\boldsymbol{x}} = f(\boldsymbol{x}, \boldsymbol{u}) \rightarrow \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}.$$

The input will depend on the current mode of the robot. In the Dock mode the inputs are computed based on the state of the compliant arm ($\boldsymbol{x}_a$), shown in Figure 4: $u_1 = f(\boldsymbol{x}_a), u_2 = g(\boldsymbol{x}_a)$. The Transport mode (Figure 3(b)) consists of two submodes: the Lead mode that uses an open loop planner and the Follow mode, where the Labmate uses information sent by the leader together with feedback information from the compliant arm to compute its input. If we use the subscript l to refer to the leader, and D is the distance between the robots, the control laws for the Lead (L) and Follow (F) submodes are:

$$\text{L} \begin{cases} u_1 = v(t), \\ u_2 = \omega(t), \end{cases} \quad \text{F} \begin{cases} u_1 = \hat{v}_l \cos(\hat{\theta}_l - \theta) + f(\boldsymbol{x}_a), \\ u_2 = (\hat{v}_1/D)\, sin(\hat{\theta}_l - \theta) + g(\boldsymbol{x}_a). \end{cases}$$

The XR4000 is holonomic having three degrees of freedom and consequently three inputs ($\dot{x} = u_1$, $\dot{y} = u_2$, $\dot{\theta} = u_3$). The behavioral hierarchy diagram for the Dock mode is show in Figure 3(c). The information from the infrared sensors ($\boldsymbol{x}_s$) is used by the XR4000 in the Approach submode: $u_1 = f(\boldsymbol{x}_s), u_2 = g(\boldsymbol{x}_s), u_3 = h(\boldsymbol{x}_s)$. The Transport mode is similar to those in other robots (see Figure 3(b)), but the definitions of the submodes are different. The Lead mode (L) uses a planner and the Follow mode (F) uses the

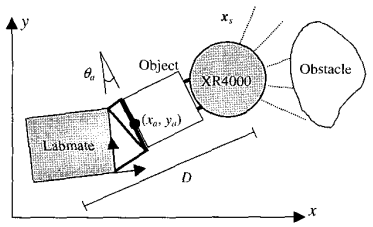

Figure 4: Diagram of the robots carrying an object - the states of Labmate's arm ($\boldsymbol{x}_a = [x_a, y_a, \theta_a]^T$) and of XR4000's IR sensors ($\boldsymbol{x}_s$) are shown

information sent by the leader:

$$\text{L} \begin{cases} u_1 = v_x(t), \\ u_2 = v_y(t), \\ u_3 = \omega(t), \end{cases} \quad \text{F} \begin{cases} u_1 = k_1(x^d - x) + f(\hat{\boldsymbol{x}}_a), \\ u_2 = k_2(y^d - y) + g(\hat{\boldsymbol{x}}_a), \\ u_3 = k_3(\theta^d - \theta) + h(\hat{\boldsymbol{x}}_a). \end{cases}$$

The terms x^d, y^d, θ^d are set points that depend on the task. For example, when the Labmate is leading the task: $x^d = \hat{x}_l + D\cos\theta$, $y^d = \hat{y}_l + D\sin\theta, \theta^d = \hat{\theta}_l$.

The Scout is non-holonomic, having the same state equations as the Labmate, but it is not used directly in the manipulation of the object. Consequently, it does not have a Dock mode and the controller in its Transport mode can be more flexible because its position relative to the leader can vary during the task execution. In the experiment presented in this paper, the Scout uses a planner when leading and sets its velocity to be equal to the leader's velocity in the Follow mode:

$$\text{Lead} \begin{cases} u_1 = v(t), \\ u_2 = \omega(t), \end{cases} \quad \text{Follow} \begin{cases} u_1 = \hat{v}_l, \\ u_2 = \hat{\omega}_l. \end{cases}$$

4 Results

Three different experiments are presented in this paper. In all experiments, the XR4000 and the Labmate cooperate to carry a box (Figure 5), but different scenarios in each experiment demonstrate several features of the architecture. The graphs of Figures 6, 7 and 10 show the trajectories executed by the robots in each experiment with data acquired from odometry. The numbers inside the graphs indicate the initial positions and the points where the leadership has been changed. Each robot is indicated by a letter (X-XR4000, L-Labmate, S-Scout).

Before beginning the transportation, the two robots must coordinate themselves to get the box. This is called the Dock mode of the cooperation. The XR4000 uses its infrared to approach the box and deploy its arm at the correct position. The Labmate uses feedback information from the compliant arm to hold the box. For now, docking is in one dimension (the robots

Figure 5: Two robots carrying a box

and the box are aligned), and the Labmate waits until the XR4000 finishes before starting its own docking.

The first experiment, shown in Figure 6, demonstrates the leader resigning its leadership. The XR4000 begins leading (0X), followed by the Labmate (0L), until it detects an obstacle using its infrared sensor (1X). Then it sends a control message resigning the leadership to the Labmate. The new leader moves backwards in a curvilinear trajectory (from 1L to 2L), returning the leadership to the XR4000 (2X) when it finishes its plan. This experiment shows that, instead of trying to avoid the obstacle locally, which is difficult to do while carrying a box in cooperation, the XR4000 offers the leadership to the Labmate, that takes it and modifies the trajectory. In this case, the modification is a simple open loop reversal with a turn.

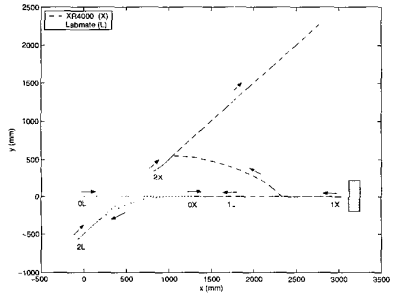

Figure 6: Experiment 1 - the XR4000 resigns the leadership (1X) and receives it back (2X)

The second experiment, shown in Figure 7, demonstrates the leadership request process. The Labmate begins leading by going backwards in a curvilinear trajectory (from 0L to 1L). The XR4000 begins following (0X) and requests the leadership when its infrared sensor detects an obstacle in its way (1X). After moving to avoid the obstacle, the XR4000 (2X) returns the leadership to the Labmate (2L) that leads until the end of the task. The leadership change here is very important because the leader is not aware of the obstacle in the path of the follower.

Figure 8 shows a discrete state diagram for Experiment 2. The arrows between the robots are the control and data messages exchanged and the links among the states in a robot are state transitions. To simplify

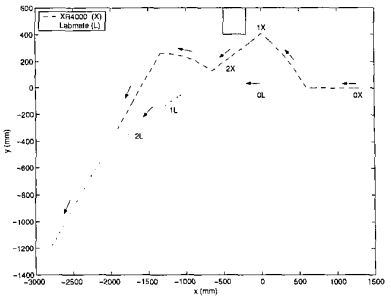

Figure 7: Experiment 2 - the XR4000 requests the leadership (1X) and returns it (2X) to the Labmate

the diagram both the acknowledgment messages and the final states are not shown. The diagrams for the other experiments are very similar to this one, meaning that it is possible to perform different actions by simply changing the role assignment sequence and the control message flow.

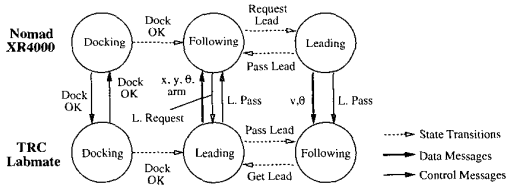

Figure 8: State diagram for Experiment 2

Figure 9 shows the modes of each robot as a function of time during the execution of Experiment 2. As shown in Figure 3(c), the XR4000's Dock mode can be divided in four submodes: Approach (A), Deploy (D), Lift (L) and Wait (W). The closeup view in Figure 9 shows a detail of a state transition. State transitions do not occur simultaneously in the two robots: there is a small delay due to communication. In this specific case, at time t_a the XR4000 sends a message requesting the leadership. The Labmate receives the message at time t_b, changes its state and sends a message to the XR4000 passing the leadership. The XR4000 only changes its state when this confirmation message arrives at time t_c. The interval $t_c - t_b$ is equal to approximately 0.03 seconds.

In the third experiment, shown in the graph of Figure 10, the Scout is used in the cooperation as a remote sensor for the Labmate. At the start position, the robots are aligned: the XR4000 is in the front (0X), the Labmate in the middle (0L) and the Scout in the back (0S). After docking, the Labmate starts leading, moving backwards until the Scout detects an obstacle using its sonar (1S). Then it requests the leadership, moves to the front (2S) and returns the leadership to the Labmate (2L), that finishes the task making a curve to avoid the obstacle.

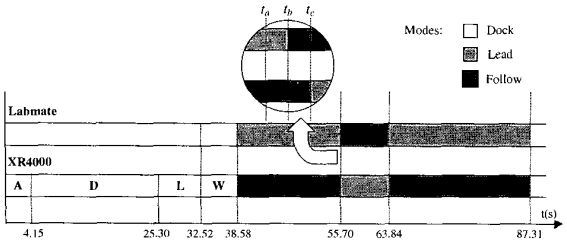

Figure 9: Time chart for Experiment 2, with a closeup view of a state transition

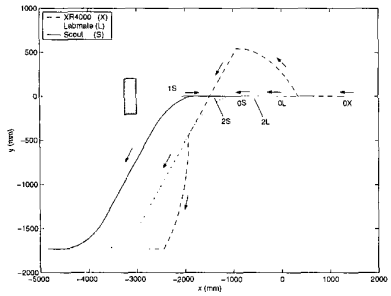

Figure 10: Experiment 3 - the Scout requests the leadership (1S) and returns it (2S) to the Labmate

5 Conclusion [1]

We presented an architecture for tightly coupled multi-robot cooperation, and the use of communication protocols and control algorithms in tasks involving grasping, manipulation and transportation of objects in an unstructured environment. A key aspect of this architecture is a mechanism for dynamically assigning roles. In particular, this allows different robots to assume the role of a leader, while others are able to adopt follower behaviors. Experimental results demonstrated the flexibility of the architecture and its performance in several typical scenarios.

Our future work is directed toward using a high-level language to formally describe the hybrid nature of the control system. We are currently using CHARON, a tool for modeling and analyzing hybrid systems [1]. Each robot is an agent, and the team of multiple coordinating robots is modeled as a parallel composition of agents. By modeling the behavior of each agent as a sequential and/or hierarchical composition of modes, we can capture the hierarchy and concurrency in such systems, while explicitly modeling the effects of communication between the robots. An important direction of this research is the study of the effect of communication protocols and reliability on system performance and robustness.

[1] Acknowledgments: Luiz Chaimowicz is supported by CAPES Foundation - Brazil and Mario Campos by CNPq.

References

[1] R. Alur, R. Grosu, Y. Hur, V. Kumar, and I. Lee, "Modular specification of hybrid systems in charon," in *Proc. of the 3rd Int. Workshop on Hybrid Systems: Computation and Control*, pp. 6-19, 2000.

[2] R. Brown and J. Jennings, "A pusher/steerer model for strongly cooperative mobile robot manipulation," *Proc. of IEEE/RJS IROS*, pp. 562-568, 1995.

[3] Y. U. Cao, A. S. Fukunaga, and A. B. Kahng, "Cooperative mobile robotics: Antecedents and directions," *Autonomous Robots*, v. 4, pp. 1-23, 1997.

[4] J. P. Desai, V. Kumar, and J. Ostrowski, "Control of changes in formation for a team of mobile robots," in *Proc. of the IEEE ICRA*, pp. 1556-1561, 1999.

[5] B. R. Donald, L. Gariepy, and D. Rus, "Distributed manipulation of multiple objects using ropes," in *Proc. of the IEEE ICRA*, pp. 450-456, 2000.

[6] Y. Hirata and K. Kosuge, "Distributed robot helpers handling a single object in cooperation with a human," *Proc. of the IEEE ICRA*, pp. 583-588, 2000.

[7] O. Khatib, K. Yokoi, K. Chang, D. Ruspini, R. Holmberg, A. Casal, and A. Baader, "Force strategies for cooperative tasks in multiple mobile manipulation systems," in *Proc. of the 7th Int. Symposium on Robotics Research*, pp. 333-342, 1995.

[8] K. Kosuge, Y. Hirata, H. Asama, H. Kaetsu, and K. Kawabata, "Motion control of multiple autonomous mobile robots handling a large object in coordination," in *Proc. of the IEEE ICRA*, pp.2666-2673, 1999.

[9] R. C. Kube and H. Zhang, "The use of perceptual cues in multi-robot box-pushing," in *Proc. of the IEEE ICRA*, pp. 2085-2090, 1996.

[10] M. J. Mataric, M. Nilsson, and K. T. Simsarian, "Cooperative multi-robot box-pushing," in *Proc. of the IEEE/RJS IROS*, pp. 556-561, 1995.

[11] L. E. Parker, "Alliance: An architecture for fault tolerant multi-robot cooperation," *IEEE Trans. on Robotics and Autom.*, v. 14, pp. 220-240, April 1998.

[12] L. E. Parker, "Current state of the art in distributed robot systems," in *Distributed Autonomous Robotic Systems 4*, pp. 3-12, Springer Verlag, 2000.

[13] T. Sugar, *Design and Control of Cooperative Mobile Robotic Systems.* PhD thesis, University of Pennsylvania, August 1999.

[14] T. Sugar, J. Desai, J. Ostrowski, and V. Kumar, "Coordination of multiple mobile manipulators," in *Proc. of the IEEE ICRA*, 2001.

[15] T. Sugar and V. Kumar, "Decentralized control of cooperating mobile manipulators," in *Proc. of the IEEE ICRA*, pp. 2916-2921, 1998.

[16] T. Sugar and V. Kumar, "Multiple cooperating mobile manipulators," in *Proc. of the IEEE ICRA*, pp. 1538-1543, 1999.