Composition of Vector Fields for Multi-Robot Manipulation via Caging

Jonathan Fink, Nathan Michael and Vijay Kumar
University of Pennsylvania
Philadelphia, Pennsylvania 19104-6228
Email: {jonfink, nmichael, kumar}@grasp.upenn.edu

***Abstract*— This paper describes a novel approach for multi-robot caging and manipulation, which relies on the team of robots forming patterns that trap the object to be manipulated and dragging or pushing the object to the goal configuration. The controllers are obtained by sequential composition of vector fields or behaviors and enable decentralized computation based only on local information. Further, the control software for each robot is identical and relies on very simple behaviors. We present our experimental multi-robot system as well as simulation and experimental results that demonstrate the robustness of this approach.**

I. Introduction

We address the problem of cooperative manipulation with multiple mobile robots, a subject that has received extensive treatment in the literature. Most approaches use the notions of force and form closure to perform the manipulation of relatively large objects [1, 2, 3]. Force closure is a condition that implies that the grasp can resist any external force applied to the object while form closure can be viewed as the condition guaranteeing force closure, without requiring the contacts to be frictional [4]. In general, robots are the agents that induce contacts with the object, and are the only source of grasp forces. But, when external forces acting on the object, such as gravity and friction, are used together with contact forces to produce force closure, we get conditional force closure. It is possible to use conditional closure to transport an object by pushing it from an initial position to a goal [5, 6]. Caging or object closure is a variation on the form closure theme. It requires the less stringent condition that the object be trapped or caged by the robots and confined to a compact set in the configuration space. Motion planning for circular robots manipulating a polygonal object is considered in [7]. Decentralized control policies for a group of robots to move toward a goal position while maintaining a condition of object closure were developed in [8].

We are interested in distributed approaches to multirobot manipulation that have the following attributes. (1) The coordination between robots must be completely decentralized allowing scaling up to large numbers of robots and large objects. (2) There must be no labeling or identification of robots. In other words, the algorithm should not explicitly encode the number of robots in the team, and the identities of the robots. Indeed the instruction set for each robot must be identical. This allows robustness to failures, ease of programming and modularity enabling addition and/or deletion of robots from the team. (3) We are interested in an approach that requires minimal communication and sensing and controllers that are based only on local information. It is often impractical for large numbers of robots to share information and to have every robot access global information. Indeed these three attributes are seen frequently in nature. As seen in Figure 1, relatively small agents are able to manipulate objects that are significantly larger in terms of size and payload by cooperating with fairly simple individual behaviors. We believe these three attributes are key to the so-called *swarm paradigm* for multi-robot coordination.

Fig. 1. Ants are able to cooperatively manipulate and transport objects often in large groups, without identified or labeled neighbors, and without centralized coordination.

In the work above, none of the papers discuss these three attributes, with the exception of [8] which does incorporate the first attribute. We note that [9] incorporates the three swarm attributes for multirobot manipulation but it makes the unrealistic assumption of point robots without considering inter-robot collisions. In this paper, we show how simple vector fields can be designed and composed to enable a team of robots to approach an object, surround it to cage it, and transport it to a destination, while avoiding inter-robot collisions. We provide the theoretical justification for the construction of the vector fields and their composition, and demonstrate the application of this approach with dynamic simulation which incorporates robot-object interactions and through experimentation.

II. Architecture and framework

Our primary interest is in performing complex tasks with large teams of distributed robotic agents. Independent of the

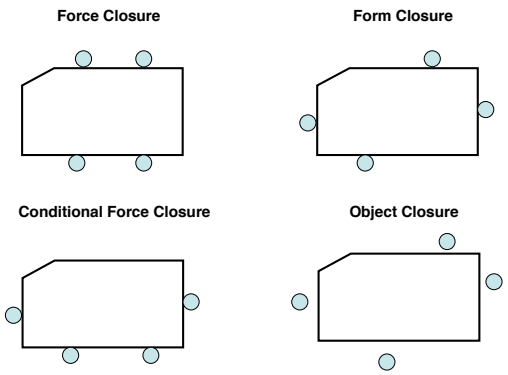


Fig. 2. The caging paradigm only requires that the object be confined to some compact set in the plane. The requirements for object closure are less stringent than form or force closure.

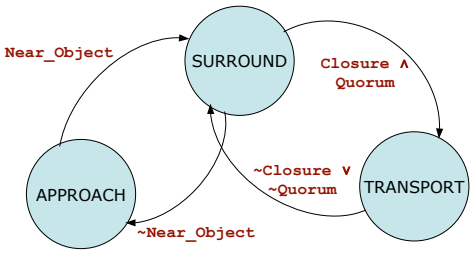


Fig. 3. Behavior Architecture. The software for each robot is identical and consists of several simple modes and the sequential composition of these modes.

specific task, scalability is of primary concern. This means: (a) Control computations must be decentralized; (b) Each robot must only rely on local information; (c) Robot controllers must be simple (the performance of a complex model-based controller does not always degrade gracefully with respect to variations in the environment). Requirement (c) essentially means that formation control based methods (for example, [3]) that rely on accurate dynamic modeling of the robots and the objects being manipulated cannot be used. In this paper, we rely on the formations that maintain closure around the object and then transport the object simply by moving along a desired trajectory while maintaining closure. See Figure 2.

A. Behaviors

Our approach to caging and manipulation of objects is summarized in the behavior architecture in Figure 3. The architecture relies on three behaviors.

1) **Approach**: The robot approaches the object while avoiding collisions with obstacles and other robots in the environment;
2) **Surround**: The robot stabilizes to a trajectory that orbits the object while avoiding collisions with other robots; and
3) **Transport**: The robot moves toward the goal configuration and/or tracks a reference trajectory that is derived from the object's reference trajectory.

As shown in the figure, the transitions between these behaviors are based on very simple conditions derived from simple sensor abstractions. If a robot is near an object, a sensor sets its `Near_Object` flag causing the robot to switch to the SURROUND mode. A `Quorum` flag is set based upon the number of neighbors within their field of view. The `Closure` flag is set when the robot network surround the object. When `Closure` and `Quorum` are both set, the robot enters the TRANSPORT mode and starts transporting the object. As the figure shows, reseting the flags can cause the robot to regress into a different mode.

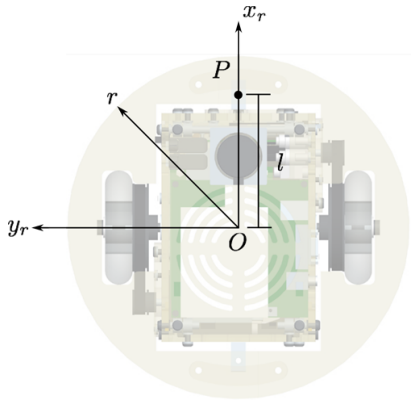


Fig. 4. A top view of the robot showing the body-fixed coordinate system. P is a reference point on the robot whose position is regulated by the vector fields.

B. Robot Model

We consider a simple model of a point robot with coordinates (x, y) in the world coordinate system. In the differential-drive robot in Figure 4, these are the coordinates of a reference point P on the robot which is offset from the axle by a distance l. We consider a simple kinematic model for this point robot:

$$\begin{aligned} \dot{x} &= u_1, \\ \dot{y} &= u_2, \end{aligned} \tag{1}$$

with the understanding that velocities of the reference point can be translated to commanded linear and angular velocities for the robot through the equations:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = \begin{bmatrix} \cos\theta & -l\sin\theta \\ \sin\theta & l\cos\theta \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix}. \tag{2}$$

It is well-known if a robot's reference point is at point P, and if r is the radius of a circle circumscribing the robot, all points on the robot lie within a circle of radius is $l + r$ centered at P. In other words, if the reference point tracks a trajectory $(x_d(t), y_d(t))$, the physical confines of the robot are within a circle of radius $l + r$ of this trajectory. In what follows, we will use the simple kinematic model of Equation (1) to design vector fields for our controllers relying on our ability to invert the model in Equation (2) for $l \neq 0$ to implement controllers on the real robot.

We will assume that each robot can sense the current relative position of the manipulated object and the state of its neighbors through local communication if necessary.

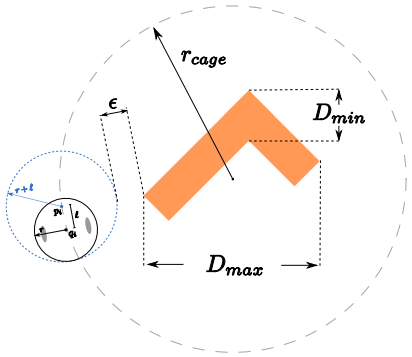


Fig. 5. An L-shaped object with circular caging shape.

C. Manipulated Object and Task

In this paper, we will consider both convex and concave objects for manipulation. But to keep the controllers simple, we will only use robot formations that are circular in shape. We will require robots to converge to circular trajectories surrounding the object. The shape of this *caging* circle is defined by three parameters. First, two parameters are defined by the manipulated object. We define the minimum diameter of the object $D_{min}(obj)$ to be the smallest gap through which the object will fit and the maximum diameter $D_{max}(obj)$ to be the maximum distance between any two points in the object. The third parameter, ϵ, is a tolerance that in turn specifies the radius of the caging circle:

$$r_{cage} = \frac{1}{2}D_{max}(obj) + (r + l) + \epsilon. \tag{3}$$

Setting ϵ too small will lead to robots bumping into the object and potentially excessive frictional forces leading to jamming. Setting ϵ too large leads to errors in following the desired trajectory. These parameters are depicted in Figure 5. For this work, we will assume all three parameters are known to all robots.

D. Command and control

While our system architecture relies heavily on autonomous agents making control decisions in a decentralized way based only on local sensing, it also depends upon a supervisory agent (which could be a human operator) that can provide task descriptions and feedback on task completion via broadcast communication. We will require that this supervisory agent can talk to all agents through broadcast commands. The agent knows the task and is able to plan trajectories for the manipulated object. However, it does not know specifics in terms of the number of robots or their initial configurations. Indeed, it does not rely on identifying individual robots in the team. This agent specifies the desired trajectory for the manipulated object and provides information on the three parameters characterizing the task ($D_{min}(obj)$, $D_{max}(obj)$, ϵ) using broadcast commands.

Our framework is inherently decentralized — individual robots make decisions on which behaviors to employ and when to take state transitions based only on local sensing. However, with a broadcast architecture it is possible for the supervisory agent to observe the global state of the system (from say an over head camera network) and provide some global information allowing the team to coordinate the robots. For example, flags could be concurrently set or reset for all the robots through broadcast commands ensuring synchronization. Note that this can be done without identifying or even enumerating the number of robots. And the amount of broadcast information is minimal and independent of the size of the team.

III. Robot Control

As shown in Figure 3, all robots are identical and the control software for each robot consists of several simple modes (behaviors) and a rule set that enables the sequential composition of these modes. In this section we will describe the distinct behaviors necessary to achieve the task of caging and manipulation and this rule set for transitions between modes.

A. Shape Control

The objective of our shape-controller and the behaviors derived from it is to surround and cage the object so that it cannot escape as the shape is moved through the workspace. Generally, the caging shape could be any enlarged approximation of the object to be manipulated but as mentioned earlier, we use a circular caging shape with radius r_{cage}.

Distributed controllers for general implicitly defined shapes are given in [10, 11]. We base our behaviors on the controller presented in [11] since it provides an orbiting component that synthesizes the Surround behavior. The basic idea is as follows.

For a desired shape given by $s(x, y) = 0$ with $s(x, y) < 0$ for (x, y) inside $\partial\mathcal{S}$ and $s(x, y) > 0$ for (x, y) outside $\partial\mathcal{S}$, we consider $\gamma = s(x, y)$ and define the navigation function

$$\varphi(q) = \frac{\gamma^2}{\gamma^2 + \beta_0}, \tag{4}$$

where $\beta_0 = R_0 - \|q\|^2$ is a representation of the world boundary. The function φ is positive-semidefinite, zero only when $s(x, y) = 0$, uniformly maximal ($\varphi = 1$ on boundary of workspace), and real analytic.

If $\psi = [0 \quad 0 \quad \gamma]^T$ such that $\nabla \times \psi$ is a vector tangent to the level set curves of φ, a decentralized control law is given by:

$$u_i = -\nabla_i\varphi_i \cdot f(N_i) - \nabla_i \times \psi_i \cdot g(T_i), \tag{5}$$

where $f(N_i)$ and $g(T_i)$ are functions used to modulate the agent's velocity towards $\partial\mathcal{S}$ and along the level sets of φ to avoid collisions with other agents via local sensing.

To construct the inter-agent terms, consider the scalar functions

$$N_{ij}(k) = \frac{(q_i - q_j)^T(-\nabla_j\varphi_j)}{\|q_i - q_j\|^k - (r_i + r_j)^k}, \tag{6}$$

$$T_{ij}(k) = \frac{(q_i - q_j)^T(-\nabla_j \times \psi_j)}{\|q_i - q_j\|^k - (r_i + r_j)^k}, \tag{7}$$

where k is a positive even number and r_i is the radius of the i^{th} element. Incorporating two switching functions

$$\sigma_+(w) = \frac{1}{1+e^{1-w}}, \tag{8}$$

$$\sigma_-(w) = \frac{1}{1+e^{w-1}}, \tag{9}$$

we can define the functions $f(N_i)$ and $g(T_i)$ to be:

$$f(N_i) = \sigma_+(N_i), \tag{10}$$

$$g(T_i) = 1 - \sigma_-(T_i), \tag{11}$$

where N_i and T_i are given by

$$N_i = \sum_{j\in\mathcal{N}_i} \left(\frac{\sigma_+(N_{ij}(2))}{\|q_i - q_j\|^2 - (r_i+r_j)^2} - \frac{\sigma_-(N_{ij}(4))}{\|q_i - q_j\|^4 - (r_i+r_j)^4} \right) \tag{12}$$

$$T_i = \sum_{j\in\mathcal{N}_i} \left(\frac{\sigma_+(T_{ij}(2))}{\|q_i - q_j\|^2 - (r_i+r_j)^2} - \frac{\sigma_-(T_{ij}(4))}{\|q_i - q_j\|^4 - (r_i+r_j)^4} \right) \tag{13}$$

Note that $f(N_i)$ and $g(T_i)$ have been constructed so that as q_i approaches q_j, $f(N_i) \to 0$ and $g(T_i) \to 0$.

Several compelling results have been shown for this controller. First, it is scalable - each controller has a computational complexity that is linear in the number of neighbors $|\mathcal{N}_i|$. Second, the system is safe ensuring that no collisions can occur between robots. Note that we do allow contact between the robot and the object — indeed it is essential to do so for manipulation. Finally, the stability and convergence properties of the controller guarantee the robots converge to the desired shape, provided certain conditions relating the maximum curvature of the boundary to the robot geometry and the number of robots are met.

In summary, the basic shape controller for agent i is given by

$$u_i = -K_N \nabla_i \varphi_i \cdot f(N_i) - \nabla_i \times \psi_i \cdot g(T_i). \tag{14}$$

The gain K_N controls the rate of descent to the specified surface relative the orbiting velocity and is used to help generate different behaviors.

B. Approach

The APPROACH behavior is characterized by a larger gain K_N on the gradient descent component of the controller to yield agent trajectories that efficiently approach the object from a distance while avoiding collisions with neighboring agents.

C. Surround

In the SURROUND mode, agents are near the desired shape and K_N is decreased so that the agents are distributed around the object. Given enough robots, this behavior will lead to object closure. For a given r_{cage} and $D_{min}(obj)$, the minimum necessary number of robots to acheive object closure is

$$n_{min} = \frac{2\pi r_{cage}}{2r + D_{min}(obj)}. \tag{15}$$

We make the assumption that there will always be at least n_{min} robots available. Additionally, for the shape controller convergence guarantees given in [11] to hold, we must maintain that no more than

$$n_{max} = \frac{\pi r_{cage}}{r} \tag{16}$$

agents attempt to surround the shape. In practice however, this is not a stringent requirement. As we will see, if the number of robots is greater than this number, because the state transitions are robust to the number of robots, the additional robots do not disrupt the caging property.

D. Transport

The controller for the TRANSPORT mode relies on the parameterization of the smooth shape $s(x, y)$ with the reference trajectory. Given a trajectory for the object $(x^d_{obj}(t), y^d_{obj}(t))$, the reference trajectory for the shape is written as

$$s(x - x^d_{obj}(t), y - y^d_{obj}(t)) = 0.$$

The vector field for the robots is otherwise unchanged. The reference trajectory adds a time-varying component to the vector field that is computed independently by each robot. The reference trajectory is computed by the supervisory agent and can be modified on the fly if needed because the broadcast architecture allows this modified trajectory to be communicated to all the robots.

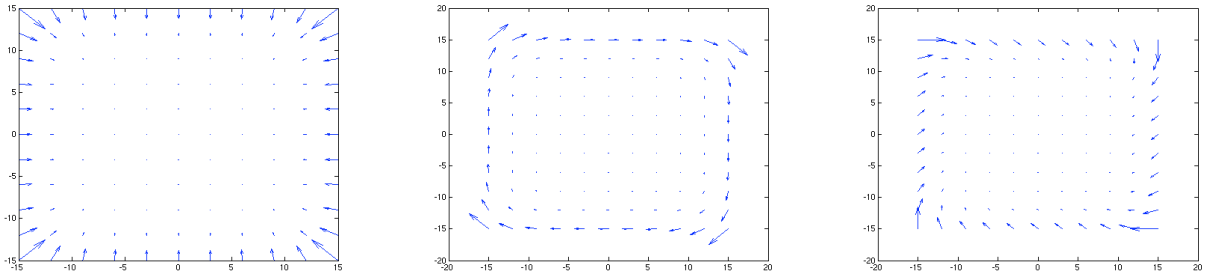
E. Composition of Behaviors

As described above, our manipulation system for multiple robots is based on a decentralized shape controller with proved stability and convergence results [10, 11]. By varying certain properties of this controller, each mobile agent in the system can operate in one of several modes including APPROACH, SURROUND, and TRANSPORT. Composition of these controller modes results in a global (non smooth) vector field that, when combined with local interactions, achieves the desired task (see Figure 6). Transitions between these modes as well as exceptions in the case of failure can be defined that allow the system to be robust while keeping individual agent decisions a function of local sensing.

In general, each agent's transition between modes will result from local observations of neighbors as well as the current distance to the manipulated object. By utilizing mode transitions that rely on locally sensed data, we can be confident this system is deployable with larger numbers of agents without any modifications.

An agent will initialize to the APPROACH mode if $d_{\text{obj}}(q_i) > d_{\texttt{near\_object}}$ (i.e. it is far from the object). As the agent approaches the desired caging shape, $d_{\text{obj}}(q_i) \leq d_{\texttt{near\_object}}$ will result in a transition to the SURROUND mode.

In the SURROUND mode, the orbiting term of the shape controller will be favored so that the robots are distributed around the object to be manipulated. Given at least n_{min} agents, this mode will converge on an equilibrium where object closure is attained. While closure is a global property of the system, it can be determined in a distributed manner by computing homology groups for the network [12, 13]. Alternately, we define an algorithm whereby closure can be locally estimated.



(a) (b) (c)

Fig. 6. The two components of the vector field, the gradient descent and rotational vector fields (Figure 6(a) and Figure 6(b), respectively). The resulting composition is shown in Figure 6(c).

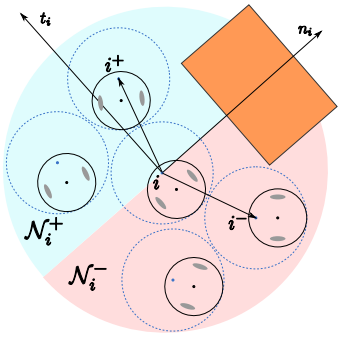


Fig. 7. Agent i's neighborhoods $\mathcal{N}_i^+$ and $\mathcal{N}_i^-$ with i^+ and i^-

1) Quorum: In order to locally define quorum, we will introduce the concept of a *forward* neighborhood $\mathcal{N}^+$ and a *backward* neighborhood $\mathcal{N}^-$ with respect to the manipulated object and the SURROUND mode. For agent i, define the normal and tangential unit vectors, $\mathbf{n}_i$ and $\mathbf{t}_i$, based on the vectors $\nabla_i \varphi_i$ and $\nabla_i \times \psi_i$ respectively. Recall that the SURROUND mode includes an approach component (in the direction $\mathbf{n}_i$) and rotation component (in the direction $\mathbf{t}_i$). Thus we can define some set of robots to be *in front of* agent i and another set *behind*. If a neighborhood N_i represents the agents within a distance $D_{min}(obj)$, then

$$\mathcal{N}_i^+ = \{j \in \mathcal{N}_i \mid 0 < (q_j - q_i)^T t_i\}, \tag{17}$$

and

$$\mathcal{N}_i^- = \{j \in \mathcal{N}_i \mid 0 > (q_j - q_i)^T t_i\}, \tag{18}$$

Furthermore, we can define agents from $\mathcal{N}_i^+$ and $\mathcal{N}_i^-$,

$$i^+ = \text{argmax}_{k \in \mathcal{N}_i^+} \frac{(q_k - q_i)^T n_i}{\|q_k - q_i\|}, \tag{19}$$

$$i^- = \text{argmax}_{k \in \mathcal{N}_i^-} \frac{-(q_k - q_i)^T n_i}{\|q_k - q_i\|}, \tag{20}$$

that will be the adjacent agents in the potential cage around the object as depicted in Figure 7.

Remembering that our agents only have the ability to observe/communicate their neighbor's state (including the value of the `quorum` variable), we propose the following update equation for $\texttt{quorum}_i$,

$$\texttt{quorum}_i = \begin{cases} 0 & \text{if } (\mathcal{N}_i^+ = \emptyset) \vee (\mathcal{N}_i^- = \emptyset), \\ n_{min} & \text{if } f(i^+, i^-) > n_{min}, \\ f(i^+, i^-) & \text{otherwise,} \end{cases} \tag{21}$$

with $f(j,k) = \min(\texttt{quorum}_j, \texttt{quorum}_k) + 1$. $\texttt{quorum}_i$ is a measure of how many robots are near agent i and in position to perform a manipulation task. Note that n_{min} is a heuristic bound on the quorum variable and there may be better choices. We shall use $\texttt{quorum}_i$, $\texttt{quorum}_{i^+}$, and $\texttt{quorum}_{i^-}$ to determine when there is object closure.

2) Closure: If there is no closed loop around the object, $\texttt{quorum}_i$ will converge to the minimum of n_{min} and the shorter of the forward/backward chain of agents. On the other hand, if there is a complete loop around the object, $\texttt{quorum}_i$ will grow as large as the imposed bound n_{min}.

We will define *local* closure to be

$$\begin{aligned} \texttt{closure}_i = &(\texttt{quorum}_i \geq n_{min}) \wedge \\ &(\texttt{quorum}_i = \texttt{quorum}_{i^+}) \wedge \\ &(\texttt{quorum}_i = \texttt{quorum}_{i^-}). \end{aligned} \tag{22}$$

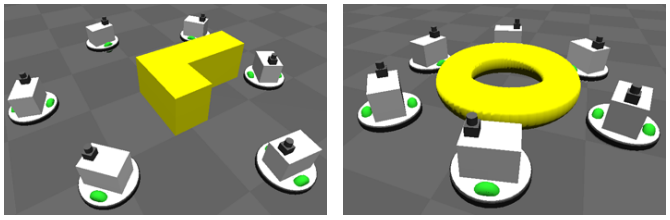
Our condition for *local* closure will coincide with *global* closure for any situation where up to $2n_{min}$ agents are used to form a cage around the object (which should not happen if agents are correctly controlling onto the specified caging shape).

When an agent estimates that *local* closure has been attained, it will switch in the TRANSPORT mode and begin attempting manipulation of the object. The `quorum` and `closure` events are defined such that they represent a kind of distributed consensus and as a result a set of manipulating agents will switch into the TRANSPORT mode in a nearly simultaneous fashion.

During manipulation an exception will occur if `closure` is lost and each agent in the system will return to the SURROUND mode to reaquire the object.

IV. EXPERIMENTAL TESTBED

The methodology for manipulation discussed in the preceeding sections was implemented in both simulation and on



(a) (b)

Fig. 8. Two images captured during GAZEBO simulations. Figure 8(a) shows six SCARAB models manipulating an "L-shaped" object in a physically correct environment. Figure 8(b) depicts the simulated robots moving a round object of the same shape as the inner-tube object shown in Figure 13(h).

hardware. We describe here the experimental testbed developed for testing scalable distributed algorithms that was used to test the proposed methodology.

PLAYER, an open source networking middleware and part of the PLAYER/STAGE/GAZEBO project [14] interfaces the distributed system and provides communication between the robots of the system. Additionally, PLAYER provides a layer of hardware abstraction that permits algorithms to be tested in simulated 2D and 3D environments (STAGE and GAZEBO, respectively) and on hardware. For this reason, all algorithm implementations discussed in Section V, both in simulation and on hardware were the same.

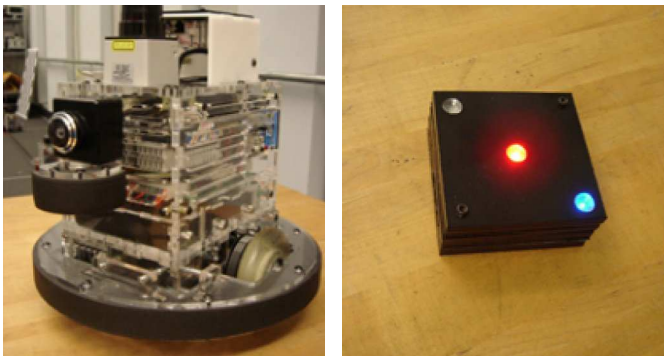
A. Simulation Environment

The open source 3D simulator GAZEBO was used to verify the algorithm. GAZEBO incorporates dynamic interactions between models via the *Open Dynamics Engine* [15]. Models of the environment of the local laboratory and hardware (discussed in Section IV-B) were reproduced in a simulated world. The robot models accurately reflect the geometric, kinematic, and dynamic descriptions of the local robots used in the hardware implementation. Frictional coefficients between the agents and the manipulated object were set to realistic values (as feasible via the *Open Dynamics Engine*).

B. Robot Platform

Our small form-factor robot is called the SCARAB. The SCARAB is a $20 \times 13.5 \times 22.2$ cm^3 indoor ground platform with a fender for manipulation with a diameter of 30 cm. Each SCARAB is equipped with a differential drive axle placed at the center of the length of the robot with a 21 cm wheel base. Each of the two stepper motors drive 10 cm (standard rubber scooter) wheels with a gear reduction (using timing belts) of 4.4:1 resulting in a nominal holding torque of 28.2 kg-cm at the axle. The weight of the SCARAB as shown in Figure 9(a) is 8 kg.

Each robot is equipped with a Nano ITX motherboard with a 1 GHz. processor and 1 GB of ram. A power management board and smart battery provide approximately two hours of experimentation time (under normal operating conditions). A compact flash drive provides a low-energy data storage solution. The on-board embedded computer supports IEEE 1394 firewire and 802.11a/b/g wireless communication.



(a) (b)

Fig. 9. The $20 \times 13.5 \times 22.2$ cm^3 SCARAB platform is shown in Figure 9(a). Figure 9(b) depicts a LED target used for localization.

The sensor models on the SCARAB are a Hokuyo URG laser range finder, a motor controller that provides odometry information from the stepper motors, and power status (through the power management board). Additionally, each robot is able to support up to two firewire cameras.

C. Other Instrumentation

The ground-truth verification system permits the tracking of LED markers with a position error of approximately 2.5 cm and an orientation error of 5°. The tracking system consists of LED markers and two overhead IEEE 1394 Point Grey Color Dragonfly cameras.

The LED marker contains three LEDs of the colors red, green, and blue. The red and blue LEDs maintain continuous illumination which are detected and tracked by the overhead cameras. The green LED flashes an eight bit pattern at a fixed interval with error checking. The pattern defines an identification number associated with each robot.

D. Algorithm Implementation

Every robot is running identical modularized software with well defined abstract interfaces connecting modules via the PLAYER robot architecture system. We process global overhead tracking information but hide the global state of the system from each robot. In this way, we use the tracking system in lieu of an inter-robot sensor implementation. Each robot receives only its state and local observations of its neighbors. An overview of the system implementation is shown in Figure 10.

V. RESULTS

The algorithms and framework presented in Sections II and III were tested through simulation and hardware implementation.

A. Simulation

We used the simulation environment to quantifiy the effectiveness of the algorithms for different object shapes and numbers of agents. The task specified was simply to manipulate the object within a distance $\epsilon + \ell$ of a waypoint. Failure was identifed by simulations that did not complete the task

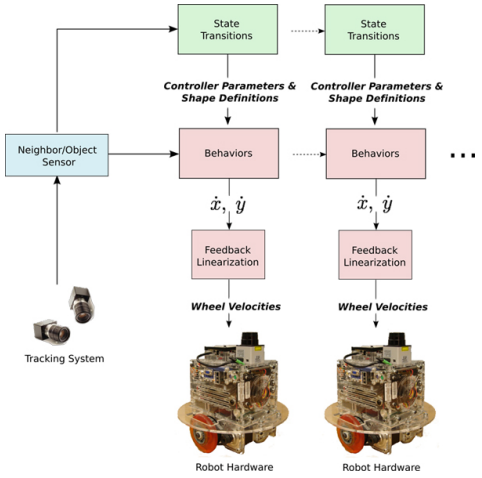


Fig. 10. Overview of the system implementation.

	Torus shape	L shape
D_{min}	1.0	0.5
D_{max}	1.0	1.0
n_{min}	4	6
n_{max}	12	12

TABLE I

OBJECT PARAMETERS (DIMENSIONS IN METERS).

within some conservative timeout period. Repeated trials in simulation were conducted for N robots manipulating both an L-shaped and torus shaped object with N ranging from n_{min} to 20. Parameters for each object are listed in Table I.

Success results from many trials in simulation are enumerated in Table II. First, it should be noted that failures only occur when $n > n_{max}$. When $n > n_{max}$ our shape controller cannot provide convergence guarantees to the specified shape which can lead to the agents orbiting a circle with radius larger than r_{cage}. With a larger caging radius, object placement within the $\epsilon + \ell$ bounds is not guaranteed and the task will not always succeed.

Representative images from a simulation trial with the L-shaped object are shown in Figures 13(a)–13(e). Note that since a trial is considered to be successful when the object is within $\epsilon + \ell$ of the goal, there is no meaningful error metric that can be reported.

B. Experimental

The methodology for manipulation was tested on the experimental testbed presented in Section IV. Four to eight SCARAB robots were instructed to manipulate a rubber inner-tube with diameter of 0.60 m and an L-shaped object with

		Success	Trials
Torus	$n_{min} \leq n \leq n_{max}$	100%	63
	$n_{max} < n \leq 20$	90%	40
L	$n_{min} \leq n \leq n_{max}$	100%	70
	$n_{max} < n \leq 20$	95%	80

TABLE II

SIMULATION RESULTS

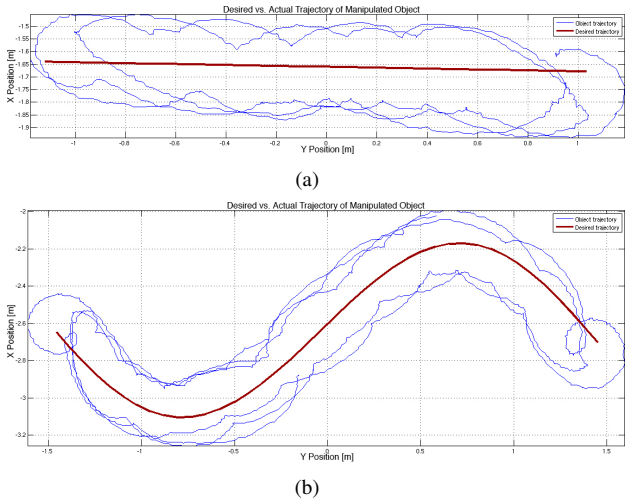


Fig. 11. Experimental Results. The linear (Figure 11(a)) and sinusoidal (Figure 11(b)) reference trajectories for the inner-tube (shown in red) and the actual trajectories traced by the geometric center of the inner-tube (shown in blue).

$D_{min} = 0.6$ m, $D_{max} = 1.2$ m. Figures 13(f)–13(j) show the robots manipulating the inner-tube object during a trial run.

Two different types of trajectories, linear and sinusoidal, were tested over ten trials. A subset of these trajectories as well as the resulting inner-tube trajectories (as defined by the center of the inner-tube) are shown in Figure 11. During experimentation no failures occurred where the object escaped form-closure. These results as well as the mean squared error (MSE) are provided in Table III.

Trajectory	Success Rate (%)	Average MSE (m)
Linear	100	0.33
Sinusoidal	100	0.38

TABLE III

EXPERIMENTATION RESULTS USING FOUR SCARABS OVER TEN TRIALS.

The robots were instructed to enclose the object in a circular shape with a radius $r_{cage} = 0.58$ m for both the linear and sinusoidal trajectories. These values (given the fender dimensions and error via the tracking system) correspond to the computed MSE.

The individual trajectories of each of the robots during the approach, surround, and transport modes for a linear trajectory are shown in Figure 12.

VI. DISCUSSION

We present a framework, the architecture, and algorithms for multi-robot caging and manipulation. The team of robots forms patterns that result in the manipulated object being trapped and dragged or pushed to the goal configuration. The controllers are obtained by sequential composition of vector fields or behaviors and enable decentralized computation based only on local information. Further, the control software for each robot is identical and relies on very simple behaviors. We present our experimental multi-robot system and simulation and experimental results that demonstrate the robustness of this approach.

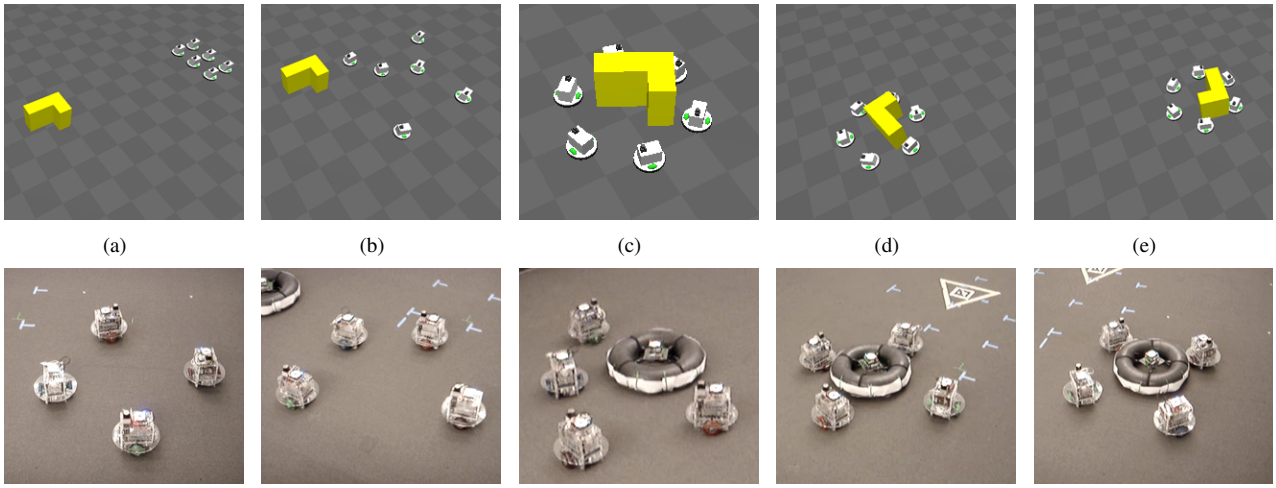


Fig. 13. A representative trial run is simulated in GAZEBO including (a) the starting formation, (b) approach, (c) surround, and (d-e) transport behaviors. (f-j) depict a similar scenario but with four SCARAB robots. The object being manipulated is a 0.60 m diameter rubber inner-tube.

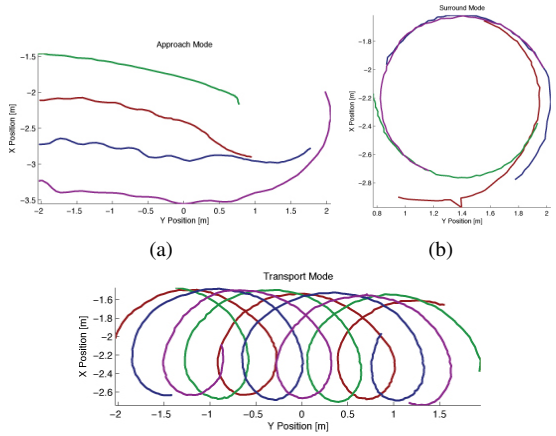


Fig. 12. Experimental trajectories of individual SCARAB robots during (a) approach, (b) surround, and (c) transport modes.

While the individual controllers and behaviors APPROACH, SURROUND and TRANSPORT have stability and convergence properties, there are no formal guarantees for the switched system, especially since each robot may switch behaviors at different times. However, our experimental results with circular objects and dynamic simulation results with objects based on hundreds of trials with more complex shapes show that this approach is robust. Though the experimental results presented in this paper were generated by transitions derived from global information, the simulated results relied on autonomous transitions and local estimates of closure. In the future we are interested in exploring exact algorithms to determine closure with without any metric information based only on local proximity sensing as in [12].

VII. ACKNOWLEDGMENT

This research was supported by: NSF grants CCR02-05336, NSF IIS-0413138, and IIS-0427313, and ARO Grants W911NF-04-1-0148 and W911NF-05-1-0219.

REFERENCES

[1] K. Kosuge, Y. Hirata, H. Asama, H. Kaetsu, and K. Kawabata, "Motion control of multiple autonomous mobile robot handling a large object in coordination," in *Proc. of the IEEE Int. Conf. on Robotics and Automation*, Detroit, Michigan, May 1999, pp. 2666–2673.

[2] D. Rus, "Coordinated manipulation of objects in a plane," *Algorithmica*, vol. 19, no. 1, pp. 129–147, 1997.

[3] T. Sugar and V. Kumar, "Multiple cooperating mobile manipulators." in *Proc. of the IEEE Int. Conf. on Robotics and Automation*, Detroit, Michigan, May 1999, pp. 1538–1543.

[4] J. K. Salisbury and B. Roth, "Kinematic and force analysis of articulated hands," *ASME Journal of Mechanisms, Transmissions, and Automation in Design*, vol. 104, no. 1, pp. 33–41, 1982.

[5] M. J. Mataric, M. Nilsson, and K. Simsarian, "Cooperative multi-robot box-pushing," in *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Pittsburgh, Pennsylvania, August 1995, pp. 556–561.

[6] K. M. Lynch and M. T. Mason, "Stable pushing: Mechanics, controllability, and planning," *The Int. Journal of Robotics Research*, vol. 16, no. 6, pp. 533–556, 1996.

[7] A. Sudsang and J. Ponce, "Grasping and in-hand manipulation: Experiments with a reconfigurable gripper," *Advanced Robotics*, vol. 12, no. 5, pp. 509–533, 1998.

[8] G. A. S. Pereira, V. Kumar, and M. F. Campos, "Decentralized algorithms for multi-robot manipulation via caging," *The Int. Journal of Robotics Research*, vol. 23, no. 7/8, pp. 783–795, 2004.

[9] P. Song and V. Kumar, "A potential field based approach to multi-robot manipulation," in *Proc. of the IEEE Int. Conf. on Robotics and Automation*, Washington, DC, May 2002, pp. 1217–1222.

[10] L. Chaimowicz, N. Michael, and V. Kumar, "Controlling swarms of robots using interpolated implicit functions," in *Proc. of the IEEE Int. Conf. on Robotics and Automation*, Barcelona, Spain, April 2005, 2487-2492.

[11] M. A. Hsieh, S. G. Loizou, and V. Kumar, "Stabilization of multiple robots on stable orbits via local sensing," in *Proc. of the IEEE Int. Conf. on Robotics and Automation*, Rome, Italy, April 2007, To Appear.

[12] V. de Silva and R. Ghrist, "Homological sensor networks," *Notices of the American Mathematical Society*, vol. 54, no. 1, pp. 10–17, 2007.

[13] A. Muhammad and A. Jadbabaie, "Decentralized computation of homology groups in networks by gossip," in *Proc. of the American Control Conf.*, New York, July 2007, To Appear.

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

[15] "Open Dynamics Engine," http://www.ode.org.