

A Carrying Task for Nonprehensile Mobile Manipulators

Atin Gupta

Wesley H. Huang

Department of Computer Science

Rensselaer Polytechnic Institute

Troy, New York 12180 USA

{guptayy, whuang}@cs.rpi.edu

Abstract

Manipulation is an essential capability for mobile robots to perform many useful tasks. Our focus has been on mobile robots with nonprehensile (i.e., nongrasping) manipulators. Here, the robots are equipped with a flat “palm” with two degrees of freedom. We have tackled the problem of carrying an object using two such mobile manipulators. Since these manipulators cannot grasp an object, each robot must support one end. However, if errors cause the separation between robots to change, the robots will drop the object. In this paper, we describe an algorithm to maintain the object contact at a nominal position on the palms by performing corrective actions. We first present analysis of the system mechanics, formulate both a centralized and a distributed algorithm for this task, and then show results of our experimental implementation.

1 Introduction

As mobile robots rise to greater prominence, they will be called upon to perform a greater variety of tasks; manipulation will be a central feature of many of those tasks. The manipulation capabilities required will include picking up, transporting, reorienting, and aligning objects. Furthermore, there will be a large range of object sizes and shapes. These tasks will occur in environments that are not engineered for robots; instead they may be in human-centered environments (e.g. office buildings, shopping malls, homes, and city streets) or in natural outdoor environments, on this or other planets. A capable mobile robotic system will require a wide variety of manipulation capabilities — the product of a versatile manipulator and an array of manipulation modes and strategies to apply to that manipulator.

We have chosen the route of simple nonprehensile (i.e. nongrasping) manipulators. In particular, our mobile robots are equipped with two degree-of-freedom (DOF) “palm” manipulators as shown in Figure 1. The palm rotates at the “wrist”, and the arm can rotate about the “shoulder”. A single robot can manipulate small objects, and two robots can cooperatively manipulate large objects. These are versatile manipulators: the palm can slide under, push, support, roll, or topple objects. Since the manipulators do not grasp the object, the object can have a wide variety of shapes and sizes.

One canonical task for mobile manipulators is to carry an object from one location to another. Previous work with

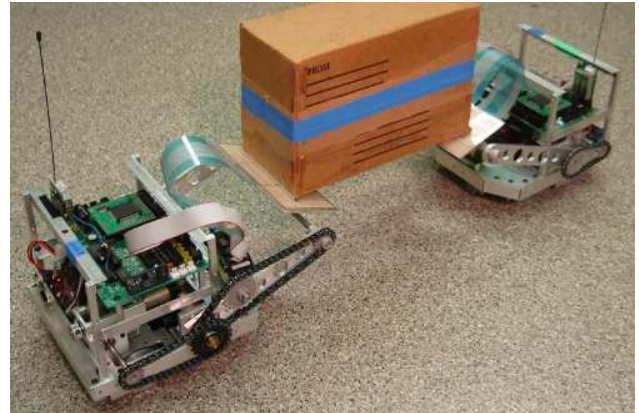


Figure 1: Two “palmbots” carry an object.

these robots (Huang and Holden [3]) addressed how a single robot can lift an object by pushing it against a wall (or another robot) and sliding the palm beneath the object. This paper addresses how two of these robots can cooperatively carry an object.

Once the object is resting on the palms, the two robots can move together to transport the object. However, because of errors or noise in the control of the robot, the separation between the two robots may change. Additionally, if the robots drive over a bump, the object may be jostled. In either case, the object can slide off one of the palms and fall to the ground. Therefore, the main goal for this carrying task is to maintain the object at a nominal position relative to the palms using feedback from tactile sensors.

The effect of a robot’s actions depends upon the mechanics of this system. Because the object can slide on the palms, predicting the object motion from the robot’s action is not trivial. This paper presents two algorithms, a centralized and a distributed algorithm, to correct any deviations from nominal object contact locations on the palms. After purveying a brief review of related work and our assumptions, we present these algorithms and demonstrate their correctness through analysis of the task mechanics. The paper concludes with the details of our hardware and results of our experimental implementation.

1.1 Related work

One common approach to mobile manipulation is to use a mobile robot to push an object on the floor. Shakey [2] was the first, and many others have followed. The more re-

cent works have focused on the mechanics or kinematics of pushing (e.g. Rus *et al.* [7] and Sudsang *et al.* [9]) or frameworks for collaborative behaviors (e.g. Kube and Zhang [5] and Mataric *et al.* [6]).

Other researchers have used six-DOF industrial manipulators mounted on mobile platforms. They take a feedback controls approach, implementing force or compliant control, and utilize redundancies due to the mobile base. This work includes Khatib *et al.* [4], Yamamoto and Yun [12], Seraji [8], and Tan and Xi [11].

The work most closely related to ours is that of Sugar and Kumar [10]. They developed a three-DOF compliant manipulator with stiffness control which they used to carry a box squeezed between the manipulators of two mobile robots. The manipulator compensates for variations in the robot separation in order to maintain the object in a stable squeeze grasp. The motion of the robots is coordinated by controllers that implement a “leader-follower” architecture.

The friction cone analysis for two-palm tasks presented in this paper is based upon work by Erdmann [1] who presented analysis and an offline planning method for manipulation tasks using two flat palms. His experimental implementation used two six-DOF manipulators. In contrast to Erdmann’s work, our present algorithm is an online algorithm which uses feedback from a tactile sensor.

1.2 Assumptions

We treat this system quasistatically, meaning that accelerations are low enough that inertial forces may be neglected. Hence gravity and frictional contact forces must balance. Our algorithm maintains the object in this quasistatic equilibrium.

The object is assumed to be an extruded polygonal shape so that the analysis may be done in the vertical plane (i.e. as viewed from the side). We presently assume that all relevant properties of the object are known: its shape, center of mass (COM), and coefficient of friction with the palms. Towards the end of our analysis, we address the range of COM locations for which our algorithm is applicable.

We have restricted the robot’s actions to tilting the palm (to a specified angle) and increasing or decreasing its velocity. The arms remain in a fixed position so that the palms’ pivot points are at equal heights above the ground. For purposes of analysis, the tactile sensor returns the exact location of the contact point on the palm. There are some differences between our actual hardware and these assumptions which we address in Section 5. We presume to know the length of the robot’s palms and their orientation, but we do not need to know the robots’ position or velocity.

2 Centralized algorithm

Our centralized algorithm takes the form of a simple policy: the global state, as determined by tactile sensor readings from both robots, triggers execution of a sequence of

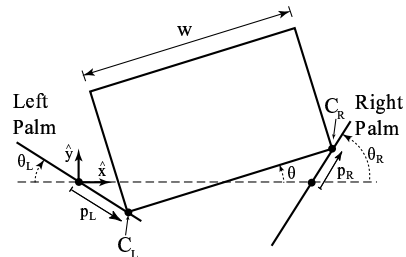


Figure 2: Geometric configuration of the system

actions. Figure 2 shows our basic notation. Though this approach is applicable to objects with any polygonal cross section, we will use the rectangular object shown to illustrate this paper.

The nominal object configuration is for both palm contacts to be at the palm pivot point, i.e., $p_L = p_R = 0$. Table 1 details our policy; there are 8 cases that are determined by the tactile sensor measurements p_L and p_R . The sequences of actions for cases 1–4 and 7–8 are designed to completely correct the object contact configuration. For cases 5 and 6, the system will be brought to one of cases 1–4.

We will use only two palm angles, θ_{shallow} and θ_{steep} , where the palms are normally held at θ_{steep} . When some action is taken, one of the palms may be “tilted up” to the θ_{shallow} orientation. We choose values for these angles so that the following two properties hold:

1. If palm separation is decreased, the contact on the shallow palm will slide up while the contact on the steep palm will stick.
2. If palm separation is increased, the contact on the steep palm will slide down while the contact on the shallow palm will stick.

The reader can easily verify that the centralized policy is correct given these properties. Cases 1 and 2 use property 2, cases 3 and 4 use property 1, and cases 7 and 8 use both properties. Cases 5 and 6 rely upon the kinematics, rather than the mechanics. (Note that an appropriate choice of two palm angles avoids wedging and jamming conditions.)

We now turn our attention to showing when these properties hold based upon the task mechanics.

3 Palmar manipulation mechanics

A contact between the object and a palm can either be sticking (ST) or sliding to the left (SL) or to the right (SR). With quasistatic stability analysis, we determine whether the contact forces (given the contact mode for both palms) can balance the gravitational force. When the left palm is shallow palm, the first property above corresponds to SL–ST, i.e., sliding left contact on the left palm and sticking contact on the right palm, the second property, to ST–SL.

We first describe a simple geometric test for quasistatic stability with two point contacts from Erdmann [1] and then use this test to establish conditions for our two properties.









Case	Contacts	Actions
1	 $p_L > 0, p_R = 0$	tilt up R, separate until $p_L \leq 0$, restore θ_R
2	 $p_L = 0, p_R > 0$	tilt up L, separate until $p_R \leq 0$, restore θ_L
3	 $p_L < 0, p_R = 0$	tilt up L, move closer until $p_L \geq 0$, restore θ_L
4	 $p_L = 0, p_R < 0$	tilt up R, move closer until $p_R \geq 0$, restore θ_R
5	 $p_L > 0, p_R > 0$	separate until $p_L \leq 0$ or $p_R \leq 0$
6	 $p_L < 0, p_R < 0$	move closer until $p_L \geq 0$ or $p_R \geq 0$
7	 $p_L > 0, p_R < 0$	tilt up R, move closer until $p_R \leq 0$, separate until $p_L \geq 0$, restore θ_R
8	 $p_L < 0, p_R > 0$	tilt up L, move closer until $p_L \geq 0$, separate until $p_R \leq 0$, restore θ_L

Table 1: Our policy for correcting errors in the object configuration. Note that “tilting up” refers to the bottom half of the palm; thus tilting the right palm up reduces the θ_R value.

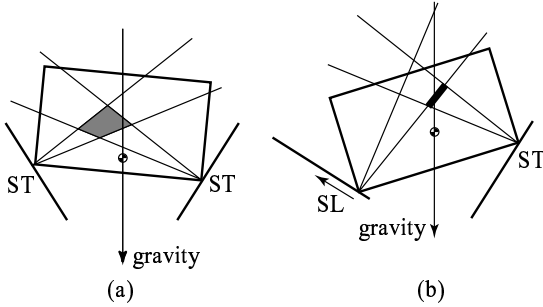


Figure 3: Geometric tests for quasistatic stability

3.1 Quasistatic stability analysis

We will assume that corners of the object (actually edges in three dimensions) are in contact with the palms, so the orientation of the palm determines the contact normal. We can determine if a configuration is quasistatically stable by first drawing the friction cones (i.e. the locus of contact forces (f_t, f_n) where $f_t \leq \mu f_n$ according to Coulomb’s law) at each contact point. For ST–ST contact, the line of action of gravity must pass through the intersection of the two friction cones as shown in Figure 3(a). For a sliding contact, only one edge of the friction cone is used: the right edge for SL and the left edge for SR. For example, Figure 3(b) shows the stability test for SL–ST.

Kinematic analysis also plays a role in this analysis. In Figure 3(b), we might try moving the palms closer together to make the right palm contact slide up the palm. Of the 9 possible contact modes, 3 are quasistatically stable, and 5 are kinematically consistent; only the SL–ST mode is both. Thus, if the palms are pushed together, the left contact will slide up the palm.

This second example also illustrates that knowledge of the mechanics is necessary to control this system. To restore the right palm contact to the center of the palm, a sequence of actions will be needed.

3.2 The SL–ST stable region

The first property of the mechanics is based on the stability of the SL–ST contact mode. In this section, we determine

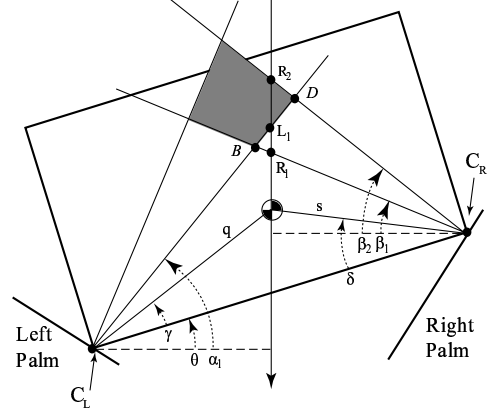


Figure 4: Geometric condition for stability of SL–ST

the shape of the stable region in the space of possible contacts on the two palms (in the p_L – p_R plane). This gives us a test to determine if a configuration is quasistatically stable for given palm angles θ_{shallow} and θ_{steep} .

We first write an expression for (the sine of) the object angle in terms of the palm angles θ_L, θ_R , contact point locations p_L and p_R , and object width w (see Figure 2 for notation):

$$\sin \theta = \frac{-p_L \sin \theta_L + p_R \sin \theta_R}{w} = \frac{z}{w} \quad (1)$$

We define $z \triangleq -p_L \sin \theta_L + p_R \sin \theta_R$ for later convenience.

The line of action of gravity intersects the lower edges of the left and right friction cones at points L_1 and R_1 (respectively) and the upper edge of the right friction cone at point R_2 as shown in Figure 4. Note that the angles introduced in this figure, α_1, β_1 , and β_2 are determined by known parameters: the palm orientations and angular friction cone widths. The y coordinates of these points are:

$$L_{1y} = q \cos(\theta + \gamma) \tan \alpha_1 + C_{Ly} \quad (2)$$

$$R_{1y} = s \cos(\delta - \theta) \tan \beta_1 + C_{Ry} \quad (3)$$

$$R_{2y} = s \cos(\delta - \theta) \tan \beta_2 + C_{Ry} \quad (4)$$

where q, s, γ , and δ are the parameters specifying the COM location as illustrated in Figure 4.

In order for this configuration to be stable under SL–ST,

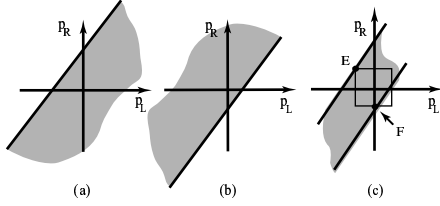


Figure 5: The stable region and operating region

the line of action of gravity must intersect the line segment BD . This implies that L_1 is to the right of B and to the left of D . This can be checked with the conditions: $R_{1y} \leq L_{1y}$ and $R_{2y} \geq L_{1y}$.

Making substitutions in the first condition, we obtain:

$$s \cos(\gamma - \arcsin \frac{z}{w}) \tan \beta_1 + z \leq q \cos(\arcsin \frac{z}{w} + \gamma) \tan \alpha_1 \quad (5)$$

Solving this inequality for z results in:

$$z \leq k_B \triangleq f(w, s, q, \gamma, \delta, \alpha_1, \beta_1) \quad (6)$$

The right hand side of this inequality is a function of the palm angles and the object geometry, COM location, and coefficient of friction, but not the contact locations p_L and p_R which appear only in z .

Substituting for z , we get:

$$p_R \leq \frac{\sin \theta_L}{\sin \theta_R} p_L + \frac{k_B}{\sin \theta_R} \quad (7)$$

which defines a half plane in the p_L - p_R space as shown in Figure 5(a).

Applying the same process to the second condition, we get:

$$p_R \geq \frac{\sin \theta_L}{\sin \theta_R} p_L + \frac{k_D}{\sin \theta_R} \quad (8)$$

where $k_D \triangleq f(w, s, q, \gamma, \delta, \alpha_1, \beta_2)$. This also represents a half plane in the p_L - p_R space as shown in Figure 5(b).

Note that the bounding edges of these half planes are parallel and have positive slopes (for $\theta_{\{L,R\}} \in [0, \pi/2]$). The intersection of these half planes is the infinite band shown in Figure 5(c).

3.3 Stability over finite palms

The points in the p_L - p_R space corresponding to finite length palms form a square centered at the origin. For the SL-ST case, we are concerned only with cases when $p_L < 0$, so we need only the left half of this square to be contained in the stable region. Because of convexity of the stable region and because its bounding edges have positive slope, we need only check the two points E and F shown in Figure 5(c).

This is a test for the first property when the left palm is the “shallow” palm. The case for the right palm is symmetric, and the second property can be demonstrated using this same procedure. Thus, this test enables us to pick values for θ_{shallow} and θ_{steep} that satisfy the two properties.

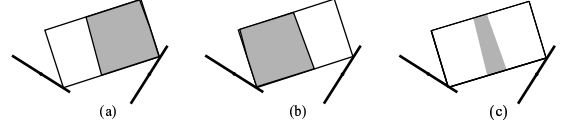


Figure 6: Range of centers of mass for stability of (a) SL-ST (b) ST-SL (c) both modes, shown on the object

3.4 Robustness to COM location

Since the location of the COM is typically not exactly known, we did further analysis to determine, for $\theta_{\text{shallow}} = 5^\circ$ and $\theta_{\text{steep}} = 15^\circ$, where the COM could be located. Our results are shown in Figure 6. For both properties to hold, the COM must be in a vertical band near the center of the object.

4 Distributed algorithm

In many respects, our distributed algorithm for this task is similar to the centralized approach. Although there is no longer a central controller that applies a policy based on the global state, the two robots are able to communicate explicitly. They may send messages to request that the other robot tilt its palm up or down (`TiltUp` and `TiltDown`) or to acknowledge the last message (`Ack`). However, the palm contact coordinate of the other robot is never shared. Both robots run the same algorithm which is shown in Table 2.

The basis of the distributed approach is that a robot can correct a negative palm contact by itself, but to correct a positive palm contact requires cooperation from the other robot. There are four cases in the distributed algorithm: Cases (a), (b), and (d) are corrective while Case (c) is assisting.

When there is a deviation on only a single robot, the algorithm is straightforward. A robot with a negative palm contact executes Case (d) to correct its own palm contact while the other robot does nothing. A robot with a positive palm contact will execute Case (a) to correct its own palm contact; it sends a `TiltUp` message which puts the other robot in Case (c) in order to assist.

When both robots have some deviation, then there must be some coordination between corrective and assisting actions. If both robots have a positive contact, then they will both execute Case (b) which coordinates their corrective actions. If both robots have a negative contact, then they will individually correct their own contact.

If one robot has a positive contact and the other, negative, then whichever robot first detects the contact deviation will correct its contact first. If that contact is positive, then the first robot will execute Case (a) and the other robot will execute Case (c) to assist. After the positive contact has been corrected, the other robot will execute Case (d) to correct its negative contact. If the negative contact is detected first, the robot will execute Case (d) to correct its own negative contact and will then execute Case (c) to assist the other robot which executes Case (a). The steps in the algorithm that wait for messages ensure this coordination.

If $p > 0$		else ($p \leq 0$)	
Send TiltUp Wait for message		Check for message	
		If message==TiltUp	otherwise
message==Ack	message==TiltUp		If $p < 0$
Move away until $p \leq 0$	Send Ack Wait for Ack Move away until $p \leq 0$ Tilt Up	Tilt Up	Tilt Up
Send TiltDown	Send TiltDown Wait for TiltDown Tilt Down	Send Ack Wait for TiltDown Tilt Down	Move closer until $p \geq 0$ Tilt Down
(a)	(b)	(c)	(d)

Table 2: Distributed algorithm

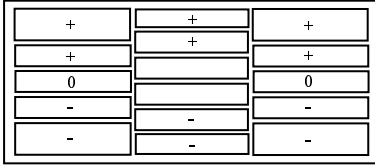


Figure 7: Tactile sensor layout. Each rectangle is a switch that is either on or off. If any of the switches labeled “+” are on, then $p > 0$. If any of the “-” labeled switches are on, then $p < 0$, and if either of the “0” labeled switches are on, then $p = 0$.

5 Experimental implementation

We implemented both the centralized and distributed algorithms on mobile robots. We first describe details of our hardware and then present our results.

5.1 Hardware details

Our robots are small differential drive robots, approximately the size of a 20 cm cube. They have on-board microcontroller and communicate via RF packet transceivers. Their palms are flat plates approximately 15 cm deep and 7 cm wide with membrane switch arrays that were custom made by Memtron Technologies. The membrane switches did not prove entirely satisfactory as tactile sensors, so we placed an aluminum strip on two edges of the box to improve the sensor readings. The layout of the switches is shown in Figure 7.

The object we used was a rectangular cardboard box, approximately 25 cm wide, 16 cm high, and 10 cm deep. We placed a weight inside the box to raise its mass to approximately 0.5 kg. The COM was 4 cm above the bottom edge (centered from side to side).

Our palm angles were $\theta_{\text{shallow}} = 5^\circ$ and $\theta_{\text{steep}} = 15^\circ$. These were chosen to be well within the stable regions that guarantee the two properties of the mechanics but shallow enough that our system would have enough time to react to contact deviations. Some amount of difference between the two angles is important to guarantee robustness to variations in the friction between the object and the palms.

The two main differences between the model of our sys-

tem and the actual robots are the discretization of the sensor information and the fact that the palm pivot is offset slightly from the palm.

5.2 Results

Figure 8 shows data collected from one run of our system using the distributed algorithm. In this run, the robots carried the object for a distance of approximately 1.5 m over a period of 25 seconds. Only the first 6.5 seconds of the run are depicted.

The robots executed 6 corrective actions in this time period. In Correction A, the right robot corrected a positive contact with the assistance of the left robot. In this correction, the right robot detects a positive contact and enters Case (a). It sends a message to the left robot which enters Case (c) and tilts its palm up (i.e., decreases the palm angle). Upon receiving the acknowledgment from the left robot, the right robot speeds up to increase separation until its palm contact reaches 0. The right robot resets its commanded velocity to the nominal value and sends a message to the left robot to tilt its palm down.

In Corrections B–D, the right robot corrects its own negative contact without involving the left robot. The right robot has not yet corrected its negative contact in Correction D when the left robot sees a negative contact in Correction E. During this time, both robots are independently trying to correct their own negative contact. The right robot corrects its deviation first, followed shortly by the left robot.

The right robot then sees a positive contact which is Correction F. It corrects this with the assistance of the left robot. The left robot gets a negative contact almost immediately at the start of this correction. However, it must first complete its role in assisting the right robot with Correction F. When this is complete, the left robot starts Correction G to correct its negative contact.

5.3 Discussion

We found that the distributed algorithm performed better than the centralized algorithm, primarily due to the decreased latency. The centralized approach requires each

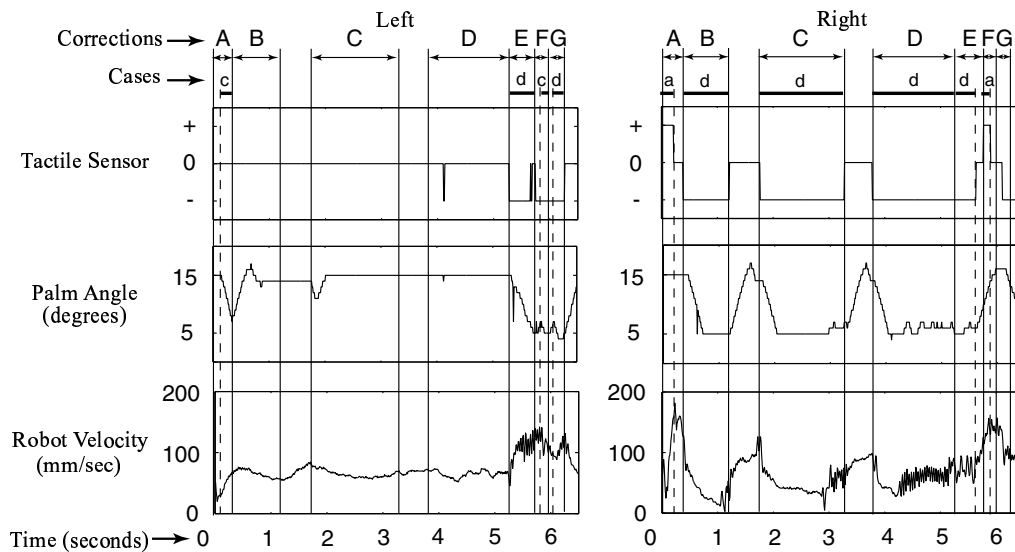


Figure 8: Data from one trial run using the distributed algorithm. Recall that tilting a palm up changes the palm angle from 15° to 5° . The robots were moving from left to right, so to decrease separation, the right robot must slow down whereas the left robot must speed up, and vice versa for increasing separation.

robot to transmit its contact state to the central controller (a desktop PC) and then for commands to be transmitted back to the robots. Under the distributed approach, robots can start reacting to contact deviations immediately and communicate directly with each other.

6 Conclusions

In this paper, we have presented an algorithmic approach for two mobile robots equipped with nonprehensile “palm” manipulators to cooperatively carry an object. By definition, these nonprehensile manipulators cannot grasp the object, so they must rely upon an understanding of the task mechanics in order maintain the object in a nominal contact configuration as the robots move. Based on analysis of the task mechanics, we devised both a centralized and a distributed algorithm for this task. The centralized algorithm takes the form of a policy that maps the global state to a sequence of actions designed to correct any deviation. The distributed algorithm is a transformation of this policy designed so that each robot can operate independently, but coordinate with the other robot (through explicit communication) when necessary.

We have successfully implemented and tested both algorithms; we found that the distributed algorithm works better primarily because there is less overhead — the centralized algorithm must receive data from the robots to determine the global state and then send commands to both robots before any corrective actions can occur.

In our future work, we plan to expand the scope of this algorithm by allowing curved robot paths and a broader class of objects. We also hope to formally expand and extend the basic approach used in this paper of transforming a global policy into a distributed algorithm.

Acknowledgments

Thanks to Matthew Leotta and Jacob Gamage for their help with the robot hardware. This work was supported by the National Science Foundation through award IIS-9983642.

References

- [1] M. Erdmann. An exploration of nonprehensile two-palm manipulation. *IJRR*, 17(5):485–503, 1998.
- [2] R. E. Fikes and N. J. Nilsson. STRIPS: A new approach to the applications of theorem proving to problem solving. *Artificial Intelligence*, 2(3):189–208, 1971.
- [3] W. H. Huang and G. F. Holden. Nonprehensile palmar manipulation with a mobile robot. *IROS 2001*, pp. 1:114–119.
- [4] O. Khatib. Mobile manipulation: the robotic assistant. *Robotics and Autonomous Systems*, 26(2–3):175–183, 1999.
- [5] C. R. Kube and H. Zhang. The use of perceptual cues in multi-robot box-pushing. *ICRA 1996*, pp. 2085–2090.
- [6] M. J. Matarić, M. Nilsson, and K. T. Simsarian. Cooperative multi-robot box-pushing. *IROS 1995*, pp. 3:556–561.
- [7] D. Rus, B. Donald, and J. Jennings. Moving furniture with teams of autonomous robots. *IROS 1995*, pp. 1:235–242.
- [8] H. Seraji. A unified approach to motion control of mobile manipulators. *IJRR*, 17(2):107–118, 1998.
- [9] A. Sudsang, J. Ponce, M. Hyman, and D. Kriegman. On manipulating polygonal objects with three 2-dof robots in the plane. *ICRA 1999*.
- [10] T. G. Sugar and V. Kumar. Control of cooperating mobile manipulators. *IEEE Trans. on Rob. and Aut.*, 18(1):94–103, 2002.
- [11] J. Tan and N. Xi. Integrated task planning and control for mobile manipulators. *ICRA 2002*, pp. 1:382–387.
- [12] Y. Yamamoto and X. Yun. Coordinating locomotion and manipulation of a mobile manipulator. *IEEE Trans. on Automatic Control*, 39(6):1326–1332, 1994.