

Centralized and Distributed Algorithms for a Cooperative Carrying Task

Atin Gupta

Wesley H. Huang

Department of Computer Science
Rensselaer Polytechnic Institute
Troy, New York 12180 USA
{guptayy,whuang}@cs.rpi.edu

ABSTRACT

Many useful tasks require both mobility and manipulation — one or more mobile robots must move to a given location and perform some task with an object there. Our research has focused on mobile robots with nonprehensile (i.e., nongrasping) manipulators. Such manipulators are mechanically simpler than their prehensile counterparts and have the additional advantage of being able to manipulate a wider variety of parts. In this paper, we present algorithms to enable two mobile robots equipped with two degree-of-freedom “palm” manipulators to cooperatively carry an object. As the robots move, the object may slip on the palms, so the robots must make adjustments, lest they drop the object. We first formulate a centralized algorithm that takes the form of a global policy. The correctness of this algorithm is demonstrated through two properties abstracted from the task mechanics. We then show how this centralized algorithm can be transformed into a distributed algorithm so that each robot can operate autonomously, communicating with the other robot only when necessary to ensure successful execution of the task. We have implemented these algorithms and present our experimental results.

I. INTRODUCTION

In order for mobile robots to play greater roles in the world, they must be able to perform a wide variety of tasks. 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. Furthermore, these tasks will generally require both navigation and manipulation capabilities: a robot or team of robots will be directed to some location and instructed to pick up, transport, reorient, align, or assemble objects at that location.

We focus here on manipulation capabilities and have chosen to use simple nonprehensile (i.e. nongrasping) manipulators. An important advantage of nonprehensile manipulators is that they can manipulate objects with a

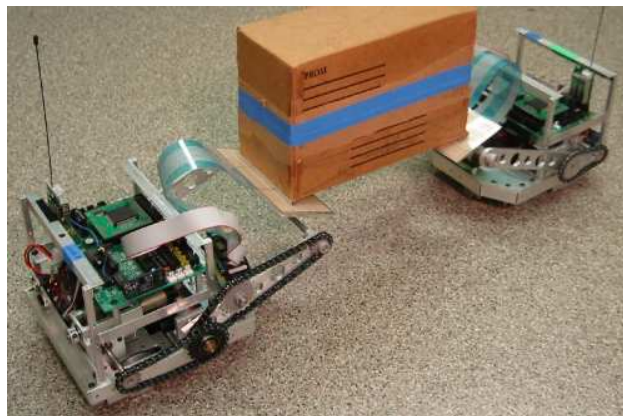


Fig. 1. Two “palmbots” carry an object.

wide variety of shapes and sizes, unlike, for example, a parallel jaw gripper which is limited by the jaw spacing.

Our mobile robots are equipped with two degree-of-freedom (DOF) “palm” manipulators as shown in Figure 1. The palm (a flat plate) rotates at the “wrist,” and the arm can rotate about the “shoulder.” A single palm can be used to slide under, push, carry, roll, or topple an object. Our previous work (Huang and Holden [1]) addressed how a single robot can lift an object by pushing it against a wall (or another robot), sliding the palm beneath the object, and lifting.

A single “palmbot” has some limitations, but these are overcome by having two or more robots cooperate. In this paper, we describe algorithms for a carrying task: two robots carry an object with each robot supporting one end of the object on its palm. We view this task and our previous work with this system as development of basic capabilities that will ultimately be combined under a framework for cooperative motion planning and execution. Other basic manipulation capabilities to be developed include reorienting and lifting strategies for two robots, so we assume that the robots start with the object resting on the robot palms as in Figure 1.

As the robots move, the object may slip on one or both of the palms. This may be due to errors or noise in the control of the robots which cause the separation between

the robots to change. Alternatively, 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 contact position on each palm by using feedback from tactile sensors.

After reviewing previous work and our assumptions for this problem, we present both centralized and distributed algorithms for this task. For the centralized algorithm, we have developed a global policy that maps the global state (the contact locations on the palms) to a sequence of actions that will return the object to its nominal contact positions. We show the correctness of this policy by establishing two properties of the underlying task mechanics and also analyze its robustness to variation in physical parameters (coefficient of friction, center of mass location, and object size). We then construct the distributed algorithm as a transformation of the global policy. The paper concludes with the details of our hardware and results of our experimental implementation.

A. Previous work

Manipulation from mobile robot platforms has been done since the earliest days of robotics. One form of mobile manipulation is for a mobile robot to push a box around on the floor. More common, however, is the use of a distinct manipulator mounted on a mobile robot. This hardware configuration, particularly with a high DOF manipulator, spurred the development of techniques to simultaneously control both the mobile base and the manipulator, generally treating them as a single redundant system.

Our task in this paper is a cooperative “carrying task,” implying that the object is completely supported by the mobile manipulators during transport. In our case, the manipulators do not have a force or form closure grasp of the object, so an understanding of the frictional contact between object and manipulator is essential.

We review previous work in each of these areas.

1) *Box pushing*: Though single robots have been used to push objects, we focus here on multiple-robot systems.

There have been a number of behavior-based approaches to multi-robot box pushing. Mataric *et al.* [2] and Parker [3] describe systems in which two robots take turns pushing the left and right sides of an elongated object. In both, the robots communicate in order to exchange sensor data and coordinate pushing. Kube and Zhang [4] use three or more robots (without any communication) to surround a box and then move towards the goal. In these approaches, there is no explicit consideration of the task kinematics or mechanics.

Spletzer *et al.* [5] and Sudsang *et al.* [6] describe approaches where mobile robots surround an object and transport it through coordinated motion. Spletzer *et al.* developed a framework for multiple robots to maintain a formation using only vision. They demonstrated three mobile robots that “trapped” a rectangular box in a

triangular formation; as the robots moved in formation, the box was transported. Sudsang *et al.* presented a motion planning algorithm that computes a sequence of paths for three or more robots that surround an object. The robots are not necessarily in contact with the object but are spaced closely enough that the object cannot escape their “grasp.” This “grasp” is maintained despite independent execution of a set paths by the robots.

Rus *et al.* [7] have examined a reorientation task for a team of mobile robots that push an object. They describe four algorithms that accomplish the same task but have different requirements in terms of communication, world models, planning, and internal state. At one extreme, their global offline algorithm requires all these facets; at the other extreme, their online asynchronous algorithm requires none. They have used information invariants (developed by Donald *et al.* [8]) to formally describe the transformation between these algorithms in an effort to characterize the information required for the task. For example, eliminating the motion planner and world model requires that certain assumptions be added to the algorithm; the information required for the task therefore remains the same. In contrast, our focus in this paper is on the transformation of a centralized algorithm to a distributed algorithm. Both algorithms contain the same assumptions about the object geometry and task mechanics. The distributed algorithm uses explicit communication for synchronizing actions in certain cases. We do not believe this task can be accomplished by a completely asynchronous algorithm.

2) *Control of mobile manipulators*: Carriker *et al.* [9] were among the first to work on control of mobile manipulators, addressing how to coordinate the motion of the mobile base and manipulator as a nonlinear optimization problem that they solved using simulated annealing. Seraji [10] combined the nonholonomic constraints of a mobile base with the kinematic constraints of a manipulator and devised a control law to satisfy both sets of constraints. Yamamoto and Yun [11] developed and implemented a control algorithm for automatically positioning the mobile base and allowing the manipulator end effector to follow an arbitrary trajectory.

Khatib *et al.* [12] focused on the dynamics of the end effector and the mobile base, demonstrating a number of single-robot tasks involving contact such as erasing a whiteboard. Brock *et al.* [13] have extended the “elastic strip” framework to mobile manipulators in order to execute reactive obstacle avoidance while satisfying the manipulator task requirements as much as possible.

3) *Carrying tasks*: There has been relatively little work done in cooperative carrying tasks. Khatib *et al.* [12] have addressed control of internal forces in cooperative manipulation using the “virtual linkage” concept. They have demonstrated cooperative carrying tasks using two Puma manipulators mounted atop Nomad mobile robots.

Perhaps most closely related to our work is that of Sugar and Kumar [14] who used two robots to carry a

box squeezed between the (nonprehensile) manipulators of two mobile robots. These manipulators are custom planar parallel manipulators with three DOF and can be actively controlled to effect its stiffness. The motion of the robots is controlled with a “leader-follower” paradigm while the manipulator stiffness is controlled to maintain a force closure grasp.

4) *Mechanics of manipulation and grasping*: The friction cone analysis for two-palm tasks presented in this paper is based upon work by Erdmann [15] who presented analysis and an offline planning method for manipulation tasks using two flat palms. His experimental implementation used two six-DOF manipulators that executed a sequence of motions open loop. In contrast to Erdmann’s work, our algorithms are online algorithms that use feedback from a tactile sensor. Though our algorithms are devised using frictional contact analysis, there is no explicit use of contact mechanics to analyze or plan for the current configuration.

More broadly, our work is an example of in-hand manipulation, each of our robots essentially acting as a “finger” in a grasp. Trinkle and Hunter [16] describe an approach to planning in the contact formation space while incorporating task mechanics. Cherif and Gupta [17] describe a multi-stage planning algorithm for reconfiguring an ellipsoidal object with three spherical fingertips. Sudsang and Ponce [18] describe in-hand manipulation using two parallel plates, one of which is fitted with a number of fingers. Rus [19] formulates an algorithm for finger tracking for in-hand manipulation of piecewise-smooth three dimensional objects.

B. Assumptions

We limit the actions of the robots so that the object is always in quasistatic equilibrium, implying that gravity and frictional contact forces must balance and that accelerations are low enough that inertial forces may be neglected. Since the object is maintained in quasistatic equilibrium, this means that the robots could be turned off at any time, and the object would be statically stable (and therefore not be dropped).

We assume that the object is an extruded planar shape so that our analysis may be done in the vertical plane, i.e., as viewed from the side. For the examples in this paper, we use a rectangular object; however, we only require that it have two edges (corresponding to vertices in the planar shape) that are in contact with the palms. For our analysis, we assume that all relevant properties of the object are known: the distance between the two contact points, the center of mass (COM) location, and the coefficient of friction with the palms. Since these quantities are often not exactly known, we examine the sensitivity of the mechanics to these quantities in Section III.

Though our robots’ manipulators have two DOF, only one is necessary to accomplish this task. We therefore keep the “arms” fixed and allow the palm to rotate

at the “wrist.” The other control that we use for this task is to increase or decrease the linear speed of a robot. The robots do not need to know their absolute linear position, but they do know the orientation of their palm. For purposes of analysis, we assume that the palm pivot is at the center of the palm. The tactile sensor measures the location of the object contact point on the palm. Our hardware differs slightly from some of these assumptions; we discuss our hardware in more detail in Section V.

II. CENTRALIZED ALGORITHM

Our centralized algorithm takes the form of a policy: the global state triggers execution of a sequence of actions. Here, the global state is determined by the tactile sensor readings p_L and p_R (from the left and right robots, respectively) which indicate the position of the contact point on the palm with respect to the palm pivot point, contacts below the pivot corresponding to negative values. See Figure 3 for notation.

The policy will correct any deviations from the nominal state, $p_L = p_R = 0$. One control available to the robots is changing the palm angles θ_L and θ_R (for the left and right robots). We will use only two palm angles, θ_{shallow} and θ_{steep} . The palms are normally held at θ_{steep} , but when some action is taken, one of the palms may be “tilted up” to θ_{shallow} . The second control is to increase or decrease the separation between the robots. This is done by temporarily increasing or decreasing the velocity of one robot from its nominal velocity.

Table I details our policy; though this approach is applicable to objects with any shape where two corners contact the palms, we will use a rectangular object to illustrate. Note that we only consider whether each contact is above, below, or at the palm pivot, so there are just 9 states in this policy.

We choose values for θ_{steep} and θ_{shallow} 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. These cases will completely correct the object contact configuration. Cases 5 and 6 rely upon the kinematics, rather than the mechanics, to bring the object contact configuration to one of cases 1–4. (Note that appropriate choice of θ_{shallow} and θ_{steep} prevents wedging and jamming conditions.)

We now turn our attention to developing a test for whether these properties hold (for given θ_{steep} and θ_{shallow}) based upon the task mechanics.

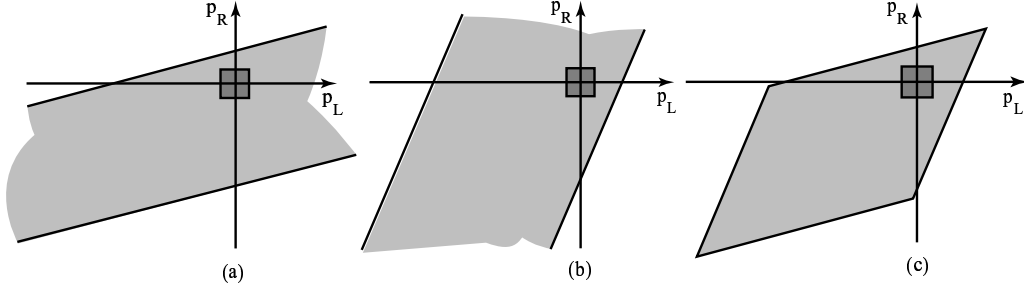


Fig. 4. (a) The stable region for ST-SL and SL-ST modes when the left palm is shallower ($\theta_L = 5^\circ$ and $\theta_R = 15^\circ$). The small darkly shaded square is the region corresponding to all possible contact configurations on our 7 cm wide palms. (b) The stable region for ST-SR and SR-ST modes when the right palm is shallower ($\theta_L = 15^\circ$ and $\theta_R = 5^\circ$). (c) The complete stable region.

section, we determine the region in the palm contact space (in the p_L - p_R plane) where the SL-ST contact mode is stable. This region must include the entire range of possible contacts between the object and the palms (the “operating region”).

For SL-ST to be stable, the vertical line through the COM must pass through intersection between the right palm friction cone and the right edge of the left palm friction cone. Using the notation in Figure 3, it must pass through the segment BD^1 . To test this condition, we write a condition on the x -coordinates of the relevant points:

$$B_x \leq L_x \leq D_x \quad (1)$$

where the world coordinate frame is taken to be at point C_L .

Applying the law of sines to triangle $\triangle BC_L C_R$, we get:

$$\frac{|BC_L|}{\sin(\theta + \beta_1)} = \frac{w}{\sin(\alpha_1 + \beta_1)} \quad (2)$$

where the length of segment $\overline{C_L C_R}$ has been replaced by w . We can then write:

$$B_x = \frac{w \cos \alpha_1}{\sin(\alpha_1 + \beta_1)} \sin(\theta + \beta_1) \quad (3)$$

Similarly, using triangle $\triangle DC_L C_R$ we find:

$$D_x = \frac{w \cos \alpha_1}{\sin(\alpha_1 + \beta_2)} \sin(\theta + \beta_2) \quad (4)$$

Since point L is directly above the COM:

$$L_x = |\vec{q}| \cos(\theta + \gamma) \quad (5)$$

By making appropriate substitutions in Equation 1 and rearranging, we get:

$$\frac{\sin(\theta + \beta_1)}{\sin(\alpha_1 + \beta_1)} \leq \frac{|\vec{q}| \cos(\theta + \gamma)}{w \cos \alpha_1} \leq \frac{\sin(\theta + \beta_2)}{\sin(\alpha_1 + \beta_2)} \quad (6)$$

Note that θ , the object orientation, is the only variable in the above equation. This equation can be solved for a

¹For extremely large μ and sufficiently shallow palm angles, the intersection consists of two disjoint regions: rays from points B and D ; see Erdmann [15] for further details. In this case, the condition is: $B_x \leq L_x$ or $L_x \leq D_x$. Though not common in our analysis, we can still find a range of θ that satisfies one of these inequalities.

range of θ values. In order to relate this range to possible contact locations on the palms, we note:

$$\sin \theta = \frac{p_R \sin \theta_R - p_L \sin \theta_L}{w} \quad (7)$$

which can be rearranged into the form:

$$p_R = \frac{\sin \theta_L}{\sin \theta_R} p_L + \frac{w \sin \theta}{\sin \theta_R} \quad (8)$$

which is the equation of a line in the p_L - p_R space. Note that the slope is fixed (for fixed palm angles), and only the p_R intercept changes as θ changes. Thus, a range of θ corresponds to an infinite band in p_L - p_R space.

C. The ST-SL stable region

When $\theta_L = \theta_{\text{shallow}}$ and $\theta_R = \theta_{\text{steep}}$, property 2 requires that the ST-SL contact mode be stable. This means that the vertical line through the center of mass must pass through the line segment AD . Again, we test this condition in terms of the x -coordinates of points in the world frame:

$$A_x \leq R_x \leq D_x \quad (9)$$

By following the same procedure as above, we can determine:

$$A_x = \frac{w \cos \alpha_2}{\sin(\alpha_2 + \beta_2)} \sin(\theta + \beta_2) \quad (10)$$

We know that $R_x = L_x$ and D_x is the same as above, so Equation 9 becomes:

$$\frac{\cos \alpha_2 \sin(\theta + \beta_2)}{\sin(\alpha_2 + \beta_2)} \leq \frac{|\vec{q}| \cos(\theta + \gamma)}{w} \leq \frac{\cos \alpha_1 \sin(\theta + \beta_2)}{\sin(\alpha_1 + \beta_2)} \quad (11)$$

As before, this can be solved for a range of θ which corresponds to an infinite band in p_L - p_R space. Since the palm angles are the same as before, this band has the same slope as that for the SL-ST stable region from the previous section. The intersection of these two bands, shown in Figure 4(a), is also an infinite band with the same slope.

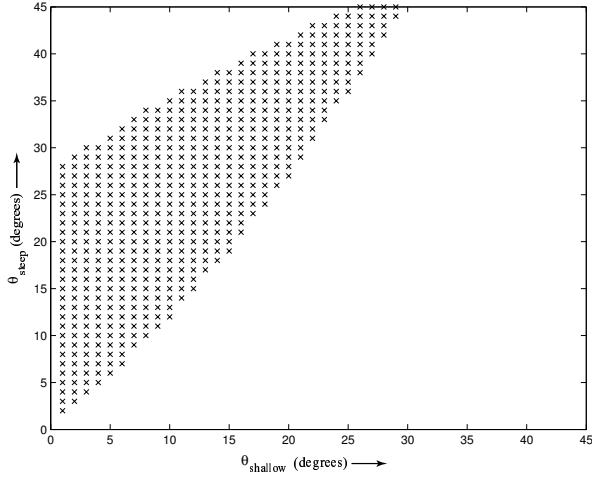


Fig. 5. The region of θ_{shallow} and θ_{steep} angles for which the two properties hold.

D. The ST-SR and SR-ST stable regions

When $\theta_L = \theta_{\text{steep}}$ and $\theta_R = \theta_{\text{shallow}}$, properties 1 and 2 require that the ST-SR and SR-ST contact modes be stable. Using a similar approach, we find that the stable region for these contact modes forms an infinite band in the palm contact space as shown in Figure 4(b).

The region of points that satisfy both properties regardless of which palm is shallow and steep is then the intersection of the bands in Figures 4(a) and 4(b). The result is the parallelogram of Figure 4(c) which we will henceforth refer to as the “stable region.” Figure 4 also shows the operating region: a square in the $p_L - p_R$ plane corresponding to all possible combinations of contact positions on the (finite length) palms. For this example, the stable region contains the operating region, so the two properties are established for $\theta_{\text{steep}} = 15^\circ$ and $\theta_{\text{shallow}} = 5^\circ$.

E. Effect of parameter variations on system stability

1) *Choice of θ_{shallow} and θ_{steep} angles:* The above analysis provides a test to determine whether the two properties hold for specific values of θ_{shallow} and θ_{steep} . Figure 5 shows the region of acceptable $(\theta_{\text{shallow}}, \theta_{\text{steep}})$ pairs, i.e., values for which both properties hold over the entire operating region. Another factor to consider is that steeper palm angles make the system more sensitive to robot separation.

We have chosen $\theta_{\text{shallow}} = 5^\circ$ and $\theta_{\text{steep}} = 15^\circ$ for our implementation. These values are reasonably far from the edge of the acceptable region but are shallow enough that small changes in robot separation do not cause large changes in palm contact positions.

2) *Robustness to COM location:* Since the location of the COM is typically not exactly known, we did further analysis to determine where the COM could be located and still have both properties hold. Our results are shown in Figure 6.

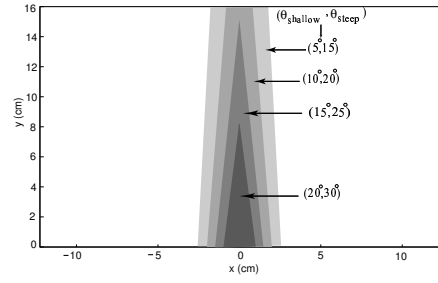


Fig. 6. Regions of acceptable COM locations for various palm angles. Note that the regions for steeper palm angles are subsets of those for shallower angles.

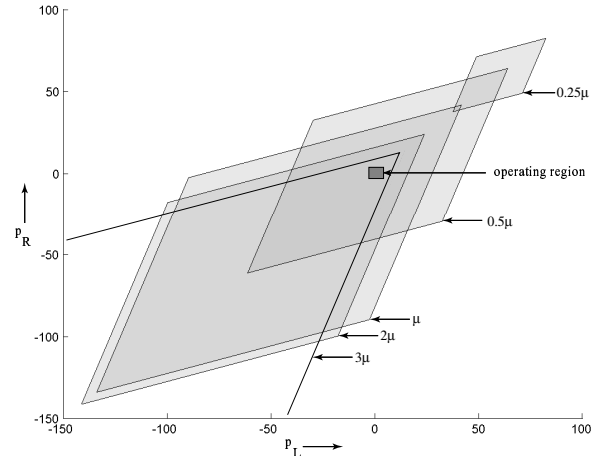


Fig. 7. Stable regions for varying coefficients of friction, here $\mu = 0.26$.

For our palm angles ($\theta_{\text{shallow}} = 5^\circ$ and $\theta_{\text{steep}} = 15^\circ$), the allowable COM locations form a vertical band in the middle of the object approximately 5 cm wide for a 25 cm wide object. This provides reasonable robustness to the COM location.

Note that that for steeper palm angles, the region of acceptable COM locations shrinks considerably.

3) *Variation in μ :* We desire a degree of robustness to variations in the coefficient of friction since it is difficult to measure precisely. Figure 7 shows the stable regions for different coefficients of friction (for fixed palm angles and COM location).

The operating region is well within the stable region for coefficients of friction from half to twice the actual value, making the system robust to a wide variation in μ .

4) *Variation in object size:* Object size may not be known exactly; for this analysis, it is only the width between the two contact points that matters. Figure 8 shows the stable regions for different object widths (for fixed palm angles and a COM in the center of the object). The operating region is well within the stable region for object widths ranging from half to twice the actual width. Note that the stable region shrinks with decreasing object width.

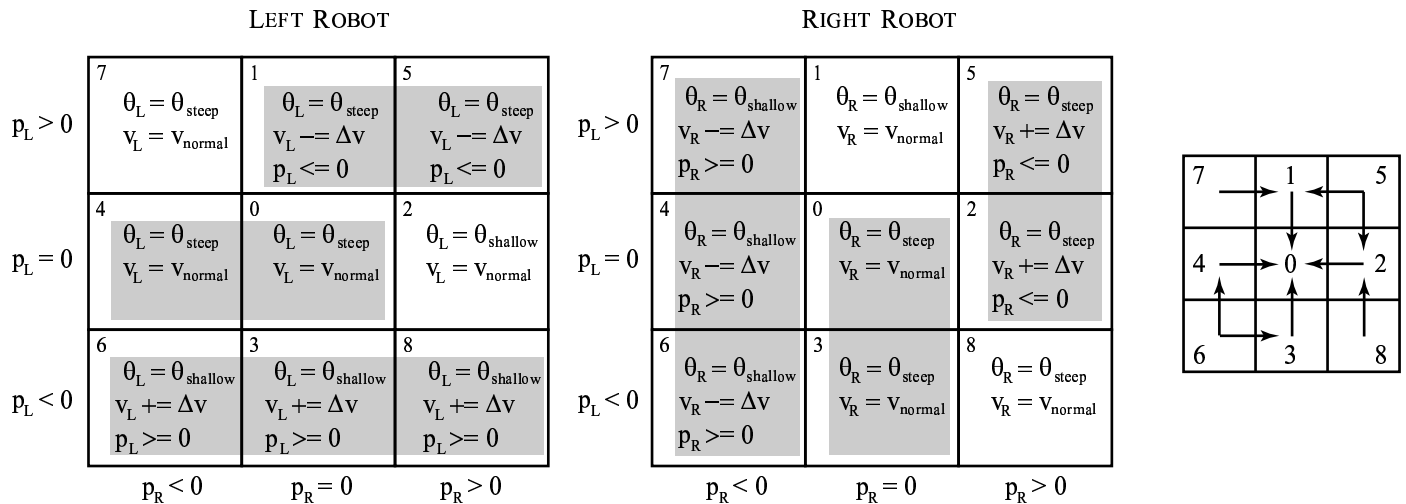


Fig. 9. The centralized policy with actions separated for the two robots. The numbers for the centralized policy cases are in the upper left hand corner of each cell. The first two lines in each cell are assignments for the palm angle and robot velocity; the last line, if present, is a condition that determines the end of that case. The shading indicates groups of identical cases. The rightmost diagram shows the transitions between the states.

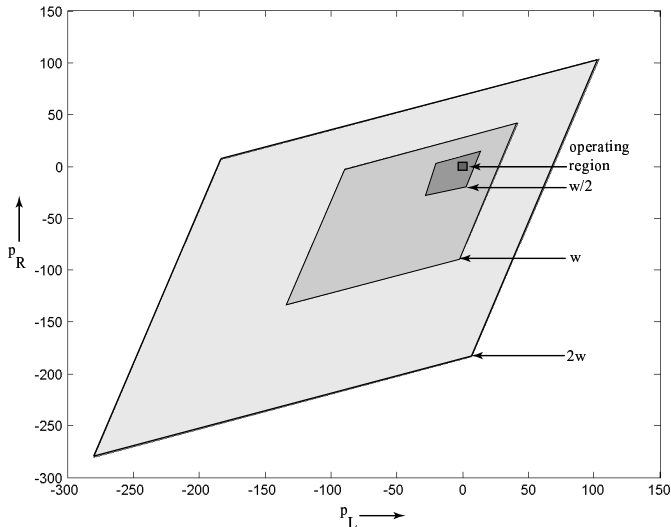


Fig. 8. Stable regions for varying object widths, here $w = 25$ cm.

IV. DISTRIBUTED ALGORITHM

Our centralized algorithm is a global policy, using the global state to select appropriate actions, and executing those actions until a change in the global state occurs. Any centralized algorithm can be trivially distributed by having each node run the centralized algorithm. This requires the nodes to exchange state and sensory information so each robot knows the global state. In addition, some synchronization mechanism may be required.

For the spectrum of possible global policies, there are two extremes. For a completely coupled global policy, the nodes must share all state and sensory information to execute the policy. For a completely decoupled global policy, each node can determine its own actions using only its own state and sensor readings. For this problem, we have something in between these extremes.

Figure 9 shows our global policy, with a few modifications, but with the actions for each robot separated. The modifications to the global policy are as follows:

- Tilting the palm up has been replaced by an explicit setting of the palm angle.
- Increasing or decreasing the separation between the robots has been replaced by an explicit setting for the robot velocity. We assume that the robots move from left to right with positive velocity to the right, and we have chosen how the separation should be changed (i.e., whether one robot should speed up or the other should slow down)
- Cases 7 and 8 in the global policy are designed to correct the contact on both palms. Here, we have chosen to correct only one palm contact which causes case 8 to be followed by case 2, case 7 by case 1.
- Case 6 tilts both palms up; this can be considered an optional step in the global policy.
- Instead of restoring the palm angles at the end of each case, this is done by case 0, the goal case.

Note that the global policies for the left and right robots are reflections of each other, with the exception of sign changes for increasing versus decreasing velocity. This indicates that both robots can run (essentially) the same distributed algorithm.

A. Transformation of the global policy

In the remainder of this section, we describe how this global policy can be transformed into our distributed algorithm (shown in Table II). Our goal is to create a distributed policy which uses minimal communication between the robots. To be concrete, we will describe this transformation in terms of the left robot.

First, we observe that the $p < 0$ case is decoupled, i.e., a robot that observes a negative palm contact should

TABLE II
DISTRIBUTED ALGORITHM (LEFT ROBOT)

Check palm contact			
$p_L < 0$	$p_L = 0$	$p_L > 0$	
$\theta_L = \theta_{\text{shallow}}$ $v_L + = \Delta v$ wait for $p_L \geq 0$	$v_L = v_{\text{normal}}$ If message (==TiltUp) $\theta = \theta_{\text{shallow}}$ Send Ack Wait for message (==TiltDown or ==Ack) $\theta_L = \theta_{\text{steep}}$	$\theta_L = \theta_{\text{steep}}$ Send TiltUp Wait for message	
		if message==Ack	if message==TiltUp
		$v_L - = \Delta v$ Wait for $p \leq 0$ Send TiltDown	$v_L - = \Delta v$ Wait for $p \leq 0$ Put back TiltUp
(a)	(b)	(c1)	(c2)

take the same action regardless of the other robot’s state. This is indicated by the shaded row for the left robot when $p_L < 0$. This establishes case (a) in Table II.

When $p_L = 0$, the appropriate action depends upon the right robot’s state. However, two of the three cases (0 and 4) are the same, and in case 4, the right robot will take actions that will result in case 0. (See the state transition diagram in Figure 9.) Therefore, the case 0 and 4 actions are the default for $p_L = 0$, and we require explicit communication for case 2. We will assume that when $p_R > 0$ a message (TiltUp) will be sent to indicate the start of this case 2. Since the left robot cannot detect the condition for ending case 2 ($p_R \leq 0$), a second message is required to end it. This establishes case (b) in Table II.

Since the distributed algorithm is the same for both robots, we have just assumed that when $p_L > 0$, a TiltUp message is sent to the other robot. The global state may be in case 7, 1, or 5, and the last two have the same action. If the global state is in case 7, then the right robot will take action that results in a transition to case 1, so case (a) in the distributed algorithm can safely ignore the message until its palm contact is corrected, resulting in a global state of case 1. Regardless of when the global state satisfies case 1, the right robot will then process the TiltUp message while executing case (b) of the distributed algorithm. The right robot tilts up its palm, acknowledges the message, and waits for a termination message (in this case, a TiltDown message). This establishes case (c1) in Table II.

One other possibility remains: if the global state is in case 5, then both robots send a TiltUp message to each other. When this occurs, neither robot tilts its palm up, and the robots decrease their separation until either $p_R = 0$ or $p_L = 0$, implying the global state is either case 1 or 2. Whichever robot detects a zero contact position first should execute case (b) as is appropriate for global state in case 1 or 2. This is achieved by the robot “putting back” the TiltUp message on the front of its message queue. When the other robot reaches $p = 0$, it too will force execution of case (b). This results in both robots sending each other an Ack message, thereby terminating

case (b). The global state at this point is now is case 0.

B. Discussion

One of the advantages of transforming a global policy into a distributed algorithm is that it is generally easier to formulate a global policy. The transformation process we described could be automated, however there are combinatorial issues that may preclude automatic transformation for larger problems. First of all, the more robots and individual robot states that are involved, the more global states that must be considered while creating a distributed policy. Second, we have made certain choices in structuring the global policy (Figure 9) from our original global policy (Table I). Though an algorithm could make these choices, each one increases the amount of computation required.

V. EXPERIMENTAL IMPLEMENTATION

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

A. Hardware details

Our robots are small differential drive robots, approximately the size of a 20 cm cube. They have on-board microcontrollers 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 10.

The object we used for most of our experiments was a rectangular cardboard box, approximately 25 cm wide, 16 cm high, and 10 cm deep. We placed a weight inside this box to raise its mass to approximately 0.5 kg and move the COM to 4 cm above the bottom edge (still 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

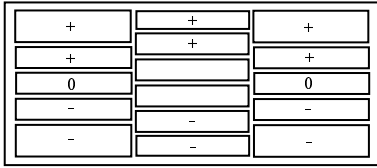


Fig. 10. Tactile sensor layout. Each rectangle is a switch that is either on or off. If any of the switches labeled “+” are on, then $p > 0$. If any of the “-” labeled switches are on, then $p < 0$, and if either of the “0” labeled switches are on, then $p = 0$.

that guarantee the two properties of the mechanics but shallow enough that our system would have enough time to react to contact deviations. The angular difference between the two angles is important to guarantee some robustness to variations in the friction between the object and the palms.

The two main differences between the model of our system and the actual robots are the discretization of the tactile sensor information and the fact that the palm pivot is offset slightly from the palm. To a certain extent, These two factors compensate for each other: with the palm pivot slightly offset from the palm, tilting the palm up or down can slightly change the contact position on the palm. However, since the region considered to be the center of the palm is approximately 1 cm wide, this does not unnecessarily trigger corrections for small contact deviations. We did not have to make any accommodation in our algorithms for these factors.

B. Results

We implemented both the centralized and distributed algorithms on our robots. Figure 11 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 shown.

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 (c1). It sends a message to the left robot which enters Case (b) 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 sends a message to the left robot to tilt its palm down, and upon entering case (b), restores its velocity to the nominal value.

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.

C. Discussion

The experimental runs were generally very successful, almost always carrying the box for 1.5 m to 3 m. The most common failure mode was when one robot veered off the straight line path due to some control error. Since the robots use no dead reckoning information and currently have no sensors to detect the other robot, there is no way for them to correct for these errors. The algorithms proved to be robust to (human introduced) external disturbances.

In another of experiment, we used a much longer box (60 cm by 11 cm by 11 cm) which was successfully transported by the robots without any modifications to the (distributed) algorithm.

We found that the distributed algorithm performed better than the centralized algorithm, primarily due to the decreased latency. The centralized approach requires each robot to transmit its contact state to the central controller (a desktop PC) and then commands must be transmitted back to the robots. Under the distributed approach, robots can start reacting to contact deviations immediately and communicate directly with each other.

VI. 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. The algorithms’ correctness is based upon properties of the task mechanics, and we have shown that these properties are robust to wide variations in the location of the COM, the coefficient of friction, and the object size.

We have successfully implemented and tested both algorithms; we found that the distributed algorithm works better primarily because there is less latency — the centralized algorithm must receive data from the robots

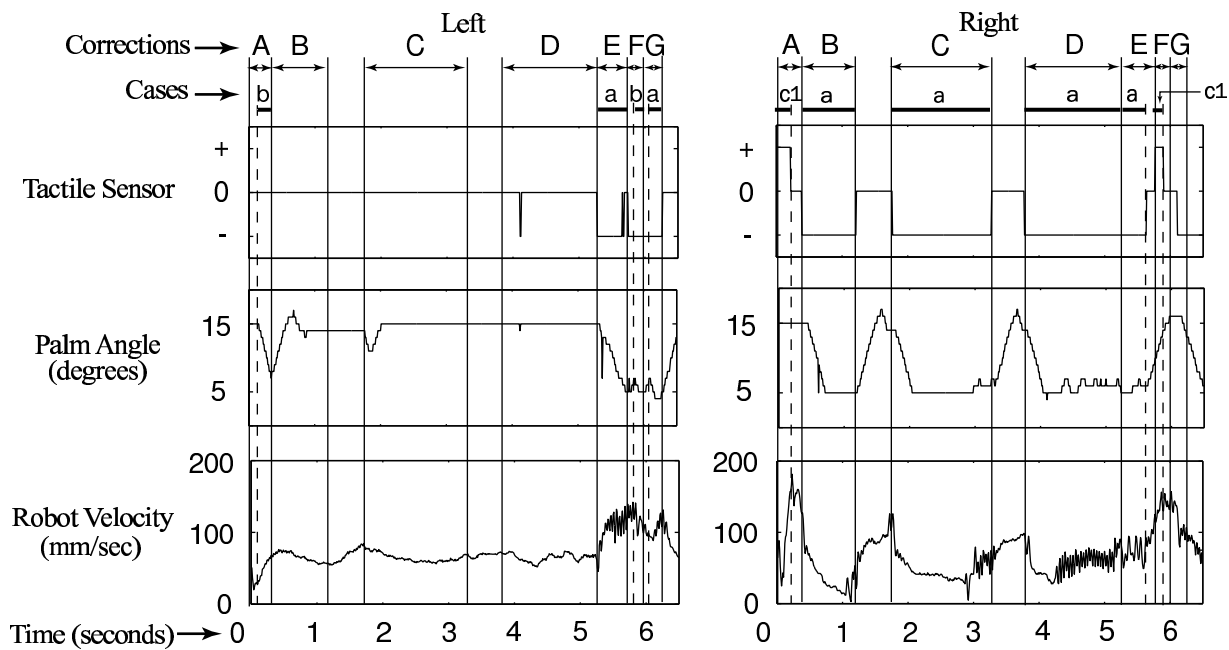


Fig. 11. Data from one trial run using the distributed algorithm. Recall that tilting a palm up changes the palm angle from 15° to 5° . The robots were moving from left to right, so to decrease separation, the right robot must slow down whereas the left robot must speed up, and vice versa for increasing separation.

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

We thank 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] W. H. Huang and G. F. Holden, "Nonprehensile palmar manipulation with a mobile robot," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 1, 2001, pp. 114–119.
- [2] M. J. Matarić, M. Nilsson, and K. T. Simsarian, "Cooperative multi-robot box-pushing," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1995, pp. 3: 556–561.
- [3] L. Parker, "Alliance: an architecture for fault tolerant, cooperative control of heterogeneous mobile robots," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 2, 1994, pp. 776–783.
- [4] C. R. Kube and H. Zhang, "The use of perceptual cues in multi-robot box-pushing," in *IEEE International Conference on Robotics and Automation*, 1996, pp. 2085–2090.
- [5] J. Spletzer, A. K. Das, R. Fierro, C. J. Taylor, V. Kumar, and J. P. Ostrowski, "Cooperative localization and control for multi-robot manipulation," in *IEEE International Conference on Robotics and Automation*, 2001, pp. 2:631–636.
- [6] A. Sudsang, F. Rothganger, and J. Ponce, "Motion planning for disc-shaped robots pushing a polygonal object in the plane," *IEEE Transactions on Robotics and Automation*, vol. 18, no. 4, pp. 550–562, August 2002.
- [7] D. Rus, B. Donald, and J. Jennings, "Moving furniture with teams of autonomous robots," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 1, 1995, pp. 235–242.
- [8] B. R. Donald, J. Jennings, and D. Rus, "Information invariants for distributed manipulation," *International Journal of Robotics Research*, vol. 16, no. 5, pp. 673–702, October 1997.
- [9] W. F. Carriker, P. K. Khosla, and B. H. Krogh, "Path planning for mobile manipulators for multiple task execution," *IEEE Transactions on Robotics and Automation*, vol. 7, no. 3, pp. 403–408, June 1991.
- [10] H. Seraji, "A unified approach to motion control of mobile manipulators," *International Journal of Robotics Research*, vol. 17, no. 2, pp. 107–118, 1998.
- [11] Y. Yamamoto and X. Yun, "Coordinating locomotion and manipulation of a mobile manipulator," *IEEE Transactions on Automatic Control*, vol. 39, no. 6, pp. 1326–1332, June 1994.
- [12] O. Khatib, K. Yokoi, K. Chang, D. Ruspini, R. Holmberg, and A. Casal, "Coordination and decentralized cooperation of multiple mobile manipulators," *Journal of Robotic Systems*, vol. 13, no. 11, pp. 755–764, 1996.
- [13] O. Brock, O. Khatib, and S. Viji, "Task-consistent obstacle avoidance and motion behavior for mobile manipulation," in *IEEE International Conference on Robotics and Automation*, 2002, pp. 388–393.
- [14] T. G. Sugar and V. Kumar, "Control of cooperating mobile manipulators," *IEEE Transactions on Robotics and Automation*, vol. 18, no. 1, pp. 94–103, February 2002.
- [15] M. Erdmann, "An exploration of nonprehensile two-palm manipulation," *International Journal of Robotics Research*, vol. 17, no. 5, pp. 485–503, 1998.
- [16] J. Trinkle and J. Hunter, "A framework for planning dexterous manipulation," in *IEEE International Conference on Robotics and Automation*, 1991, pp. 2:1245–1251.
- [17] M. Cherif and K. K. Gupta, "Planning quasi-static fingertip manipulations for reconfiguring objects," *IEEE Transactions on Robotics and Automation*, vol. 15, no. 5, pp. 837–848, October 1999.
- [18] A. Sudsang and N. Ponce, J. and Srinivasa, "Grasping and in-hand manipulation: geometry and algorithms," *Algorithmica*, vol. 26, no. 3–4, pp. 466–493, March–April 2000.
- [19] D. Rus, "In-hand dexterous manipulation of piecewise-smooth 3-d objects," *International Journal of Robotics Research*, vol. 18, no. 4, pp. 355–381, 1999.