Abstractions and Algorithms for Cooperative Multiple Robot Planar Manipulation

Peng Cheng Jonathan Fink Vijay Kumar
GRASP Laboratory
University of Pennsylvania
Philadelphia, PA 19104 USA
{chpeng, jonfink, kumar}@grasp.upenn.edu

***Abstract*— In this paper, we will study abstractions and algorithms for planar manipulation systems using two cooperating robots under uncertainties. We propose a formal framework for developing abstractions, which are simpler models of the original systems that preserve properties of interest to facilitate the development of planning and control algorithms. Our abstractions are derived from robust motion primitives that correspond to control inputs leading to system trajectories which preserve the properties of interest under uncertainties. We then use the proposed framework to construct an abstraction and design planning and control algorithms for a multiple robot cooperative manipulation system. Finally, we present experimental results to validate our approach.**

I. Introduction

It is well known that conventional approaches to robotic manipulation, where deliberative planning is augmented by feedback controllers, are difficult to implement except in the simplest of cases. This is primarily because of non smooth dynamics engendered by frictional contacts and uncertainties in the parameters governing the contact dynamics. Experiments in robotic juggling [8], locomotion [11, 25], non prehensile manipulation [35], manipulation via caging (Fig. 2) [13], and part-feeding [29] have shown that feedback controllers, behaviors or designs, which are specially designed to preserve a specific set of properties (*e.g.*, convergence to sub manifolds or limit sets), are more robust to uncertainties than those that follow optimally-planned trajectories in the full state space. Indeed, this philosophy of designing components that each drive the system to a state that satisfies a specific property is used extensively in manufacturing operations, where designers carefully structure the environment to ensure that devices like bowl-feeders [14], conveyors [1], traps [5], and pick-and-place arms work in concert to accomplish the given task. Many paradigms in robotics such as caging [7], the one-joint-over-conveyor part positioning [1], and remote-center-of-compliance assembly [12] are also illustrative of this philosophy. While these examples are arguably special-purpose solutions, they illustrate a very important point. By designing planners/controllers that drive the system to submanifolds in the state space, one can derive *abstractions* of complex processes, *i.e.*, conceptual models that are much simpler than the complex real-world system, that lend themselves to the design of *algorithms* that can reason about these abstractions and the composition of these complex processes.

We use the simple example of multi-fingered or multi-robot manipulation in the plane via caging to illustrate the role of abstractions and algorithms (Fig. 2). The modeling of multi-fingered hands or multi-robot manipulation is complicated by the fact it involves multi-body dynamics with frictional contacts. Static indeterminacy and frictional impacts introduce additional difficulty making the design of provably-correct planners and controllers impractical. However, in many manipulation tasks the main goal is to position and orient an object to some destination with a specified tolerance. Since the main *property of interest* is the geometric property of containing or enclosing the manipulated object, one is motivated to derive geometric *abstractions* for the complex, multi-dimensional dynamics problem. This is the central idea in configuration-space abstractions used to derive *algorithms* for multi-robot manipulation: motion planning algorithms for caging [33, 32], control algorithms for object closure [21], and composition of controllers for multi-robot manipulation [13]. Each robot or finger is abstracted into a geometric model. And the planning/control problem is to determine how to move/control these geometric entities to enforce geometric closure.

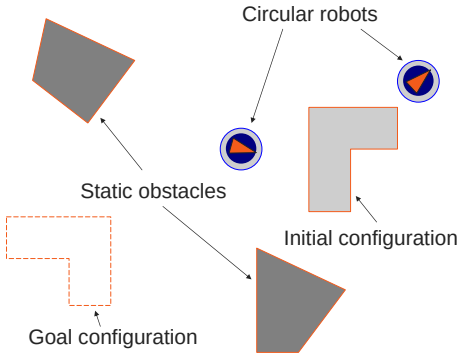

Fig. 1. A representative assembly problem.

In this paper, we will construct abstractions and design planning and control algorithms for multi-contact, planar manipulation tasks in which multiple nonholonomic mobile robots cooperate to manipulate a 2.5-dimensional object on an even, rough surface (see Fig. 1). The manipulation problem in such scenario is very challenging due to non-smooth dynamics and frictional contacts as well as uncertainties in sensing, actuation, and system parameters (*e.g.* friction coefficient and unknown support distribution). It has been studied in [19, 2, 4, 18, 33, 22]. Our work is very similar to [33, 13] in application (using circular robots to manipulate polygonal parts). However, geometric abstractions of caging are used in [33,

(a) (b) (c)

Fig. 2. Approaches to cooperative manipulation and multi-fingered grasping that rely on form or force closure [6, 23, 24, 26] are not as robust to uncertainties as *object closure*, in which the robots or fingers enclose or cage the object. Robots can approach (Fig. 2(a)), surround (Fig. 2(b)), cage (Fig. 2(c)) and manipulate or transport the object reliably using geometric abstractions associated with caging [13].

13], which require at least three robots and large operational space. Also, caging in [13] provides few guarantees on part orientation. In assembly tasks like the one in Fig. 1, it is hard to use caging to drive the part to a goal configuration within a specified tolerance in a constrained environment. The property preserved by our abstraction is neither enclosing nor caging, but to maintain contacts between two manipulation robots and the part. This idea is similar to stable pushing [18]. However, instead of preserving sticking contact between the part and a single pushing bar, we are using two robots to cooperatively manipulate the part by preserving contacts (either rolling or sliding) between both robots and part.

The remainder of the paper is organized as follows. Section II provides a formal framework of abstractions for the manipulation system. The multiple robot cooperative manipulation problem is described in Section III. Abstraction and algorithms for such system are provided in Section IV-B with a focus on abstraction. In Section V, we provide experimental results to validate the proposed approaches.

II. Abstractions of Manipulation Systems

We define a manipulation system by the tuple, $\mathcal{M} = \{f, X, U, \mathcal{P}, T\}$, where X denotes the state space, U the input space, $\mathcal{P}$ the space of (possibly time-varying) model parameters, T a finite time interval, and f the differential equation characterizing the flow of the system. To distinguish between the value of controls ($u \in U$), parameters ($p \in \mathcal{P}$) or state ($x \in X$) from the corresponding trajectories, we use the notation $\tilde{(\cdot)}$ to indicate histories or trajectories. Thus $\tilde{u}$ is the input history, while $\tilde{p}$ is the history of parameter variation and will be used to represent uncertainties. Given a control $\tilde{u} : T \to U$, a parameter history $\tilde{p} : T \to \mathcal{P}$, and an initial state $x_0 \in X$, the trajectory is given by $\tilde{x}(x_0, \tilde{u}, \tilde{p}, t) = x_0 + \int_0^t f\left(\tilde{x}(\eta), \tilde{u}(\eta), \tilde{p}(\eta)\right) d\eta, t \in T$.

We use $\tilde{X}$ to denote the set of trajectories with all possible initial states, controls, and parameter histories. We now define the *property* of interest for the system that characterizes the successful execution of a task or subtask as a polymorphic characteristic function, $\Phi : \tilde{X} \to \{0, 1\}$, which determines whether or not a trajectory of model $\mathcal{M}$ satisfies the given property. It is polymorphic (in analogy to polymorphism in object-oriented programming [3]) because, as we will see, the property function can be used to characterize either the original model or its abstraction. We can also define a subset, a collection of trajectories, $S \subset \tilde{X}$, satisfying a given property: $S = \{\tilde{x} \in \tilde{X} \mid \Phi(\tilde{x}) = 1\}$. In particular, we will be interested in the trivial property, Φ_0, that is satisfied by all trajectories

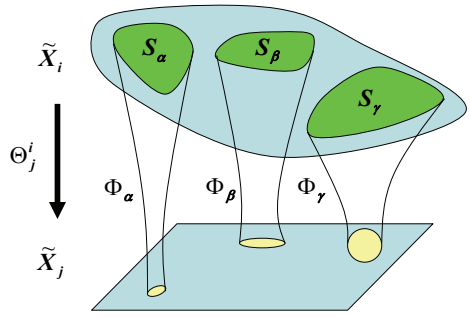

Fig. 3. The surjective map Θ_j^i maps $\tilde{X}_i$ to $\tilde{X}_j$. S_α, S_β, and S_γ are sets of trajectories preserving the properties Φ_α, Φ_β, and Φ_γ.

satisfying the system equations for the model $\mathcal{M}$. In this case, $S = \tilde{X}$. We now establish conditions under which a model $\mathcal{M}_j$ is an abstraction of $\mathcal{M}_i$ with respect to a property, Φ. We use subscript i and j to distinguish components from model $\mathcal{M}_i$ and $\mathcal{M}_j$. Thus, $x_i \in X_i$ is a state in model $\mathcal{M}_i$ and $x_j \in X_j$ is associated with $\mathcal{M}_j$. To keep matters simple, we assume the system is time-invariant and the system dynamics are characterized by a vector of constant parameters for both models and we will omit the dependence on p in the discussion in this subsection. We construct $\mathcal{M}_j$ so that the underlying state space X_j is an image of X_i under the surjective map Θ_j^i. This in turn induces a map in trajectory space as shown in Fig. 3. We say that $\mathcal{M}_j$ is a sufficient abstraction[1] of $\mathcal{M}_i$ if, for any trajectory $\tilde{x}_j(x_j, \tilde{u}_j, t)$ in $S_j \subset \tilde{X}_j$ satisfying the property Φ, there exist $\tilde{u}_i$ and $x_i \in X_i$ so that $\Theta_j^i(x_i) = x_j$ and

$$\Phi\left(\tilde{x}_j(\Theta_j^i(x_i), \tilde{u}_j, t)\right) = \Phi\left(\tilde{x}_i(x_i, \tilde{u}_i, t)\right) = 1. \qquad (1)$$

A simple example of this sufficient condition is seen in fully-actuated, six degree-of-freedom robot arms. We frequently use kinematic abstractions ($\mathcal{M}_j$) and inverse-kinematics-based algorithms to plan tasks and trajectories for the tasks because we know that computed-torque-based nonlinear feedback controllers for the real dynamic system ($\mathcal{M}_i$) can be used to realize paths synthesized by simpler kinematic controllers. In other words, these two models satisfy the sufficient condition with respect to the trivial property Φ_0.

III. Cooperative planar manipulation

A. The task

We consider the representative problem, depicted in Fig. 1, in which multiple robots are able to manipulate the object into the desired goal. The robots are position-controlled without force or contact sensors. They are able to sense the relative position and orientation of the object and coordinate via communication before manipulation. Because of latency in the network and imperfect sensing, the control during the pushing motion must be open-loop. This paradigm is typical of assembly tasks in industry where robot tasks often involve sequences of subtasks each involving sensing before the subtask, computation, followed by execution. Our goal is to design

[1]One can also define necessary abstractions using a necessary condition in a similar way but since we will not use this concept, we will not discuss this further in this paper.

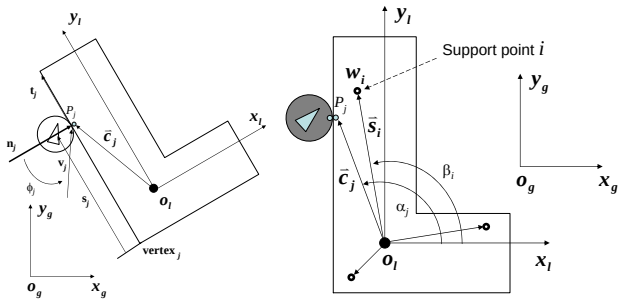

Fig. 4. Geometry and system with the inertial frame (left) and position vectors in the body-fixed frame (right). The object has three, unknown support points ($i = 1, 2, 3$) denoted by unfilled circles. The center of mass is denoted by $\mathbf{o}$, denoted by a filled circle. The contact point between the object and the robot is denoted by P_j with the inward-pointing normal $\mathbf{n}_j$.

controls for multiple robots to manipulate the part to a given configuration within specified tolerances.

Alternative approaches based on force closure require force sensors which lead to expensive and unreliable hardware. Instead we use Roomba-like nonholonomic robots that are position-controlled to follow desired trajectories.

Accordingly, we restrict ourselves to the quasi-static regime where the inertial forces are small compared to the contact forces applied by the robots. Further, we use circular robots to simplify the geometry and the algorithms required for planning and control. Because the application of more than two frictional contacts always results in static indeterminacy we only use two robots at any given time.

B. Modeling and notation

Consider the representative part shown in Fig. 4. We adopt the frictional, three-point support model from [22] but recognize that these support points can change as the part moves and their locations are unknown. The robot(s) exhibit frictional, point contact with the object. All coefficients of friction are unknown but lie within a known set. The part geometry, its inertial properties, and the location of the center of mass are known.

The weight of the part, $w = mg$, is supported by three, unknown support points S_i $(i = 1, 2, 3)$ with coordinates (x_i, y_i). The position and orientation of the part is denoted by $q = (x, y, \theta)$ and its velocity in the inertial frame is $\dot{q} = (\dot{x}, \dot{y}, \dot{\theta})$. The body-fixed frame, $x_l - y_l$, has its origin at the center of mass o_l. The jth contact with the jth robot occurs at P_j whose position vector in the body-fixed frame is $\vec{c}_i$.

The robot velocity is $\mathbf{v}_{R,j}$ while the velocity of the point P_j on the part is $\mathbf{v}_{P,j}$. The relative velocity at P_j is given by components $(v_{n,j}, v_{t,j})$ denoting the separation velocity and sliding velocity respectively:

$$v_{n,j} = (\mathbf{v}_{P,j} - \mathbf{v}_{R,j}).\vec{n}_j,\ v_{t,j} = (\mathbf{v}_{P,j} - \mathbf{v}_{R,j}).\vec{t}_j.$$

The forces on the object include the normal forces w_i, the tangential frictional forces at support S_i $(f_{s,i,x}, f_{s,i,y})$, as well as the robot-object contact force at P_j, with components $(\lambda_{n,j}, \lambda_{t,j})$ along the inward-pointing normal $\vec{n}_j$ and tangent $\vec{t}_j$. μ_s is the coefficient of surface friction while μ_c is the coefficient of friction at the robot-object contact.

C. Uncertainties

Although we use the three-point support model to predict the force distribution, we allow the support points S_i to vary. They are chosen to lie within a specified set E_s with the constraint that the center of mass falls within the support triangle.

The friction coefficients between the part and the support and between the part and robots are unknown, but they are assumed to belong to a known, compact set E_f.

The errors in sensing the position and orientation of the object/part and the errors in controlling individual robots must be modeled. The errors in positioning and orienting are denoted by E_t and E_θ. E_d denotes the errors on the relative positions of the robots. In our case, since this is related to the sensing error, $E_d = 2E_t$. We use E_v to denote the error in relative velocity.

D. Quasi-static model for planar manipulation

The non negative normal force at S_i denoted by w_i are uniquely determined from the force equilibrium in the vertical (out-of-plane) direction and the coordinates of the support points:

$$\sum w_i = w,\ \ \sum w_i \vec{s}_i = 0. \tag{2}$$

The force-balance equations (forces and moments about o_l) in the plane are:

$$W_n(q)\lambda_n\ + W_t(q)\lambda_t = w_s \tag{3}$$

where $\lambda_n = [\lambda_{n,1},\ \ \lambda_{n,2}]^T$ and $\lambda_t = [\lambda_{t,1},\ \ \lambda_{t,2}]^T$ and $w_s = \sum_{i=1}^{3} w_{s,i}$ is the resultant support wrench. The wrench matrices W_n and W_t are given by:

$$W_n = \begin{bmatrix} \vec{n}_1 & \vec{n}_2 \\ \vec{c}_1 \times \vec{n}_1 & \vec{c}_2 \times \vec{n}_2 \end{bmatrix},\ W_t = \begin{bmatrix} \vec{t}_1 & \vec{t}_2 \\ \vec{c}_1 \times \vec{t}_1 & \vec{c}_2 \times \vec{t}_2 \end{bmatrix}$$

We write the tangential sliding velocity as the difference of two non negative quantities:

$$v_{t,j} = v_{t,j}^{+} - v_{t,j}^{-}. \tag{4}$$

We can now write the following complementarity conditions [30]:

$$0 \le v_{n,j} \cdot n_j \perp \lambda_{n,j} \ge 0. \tag{5}$$

$$0 \le v_{t,j}^{+} \perp \lambda_{n,j}\mu_c + \lambda_{t,j} \ge 0 \tag{6}$$

$$0 \le v_{t,j}^{-} \perp \lambda_{n,j}\mu_c - \lambda_{t,j} \ge 0. \tag{7}$$

Note that Equations (2-7) provide a comprehensive description of the system independent of whether each contact is separating, rolling or sliding [34]. Although the uniqueness and existence properties for this set of equations has not been established for the general case, it is possible to show that under conditions of *positive-linear independence* [28], there is a unique solution. This is discussed again in the next section.

E. Practical considerations

In order to ensure the quasi-static assumption is satisfied, we must ensure that the kinetic energy of the object never exceeds the energy that can be dissipated due to friction in some small time interval. Specifically, we are concerned with errors in sensing E_t and E_θ and we want to make sure that

the kinetic energy of the object does not cause it to translate more than E_t or rotate more than E_θ. Accordingly we require

$$\dot{x}^2 + \dot{y}^2 \ll E_t g \mu_s \qquad R\dot{\theta}^2 \ll E_\theta g \mu_s. \tag{8}$$

which in turn restricts the velocity of our robots. Second, we cannot require forces that exceed the maximum frictional force or traction between the robot and the support surface.

$$\sqrt{\lambda_{n,i}^2 + \lambda_{t,i}^2} < t_{\max} \tag{9}$$

The robot sensors and controllers, their dynamic properties and the properties of the object will impose further constraints. To ensure robustness to communication latencies and delays we assume that all robots coordinate their execution but do not exchange state information during the manipulation task. The robots used for experiments are approximately 8 Kg. We choose an L-shaped object for manipulation whose mass is around 2.5 Kg. The coefficients of friction are $\mu_s = 0.08 \pm 0.02$ and $\mu_c = 0.6 \pm 0.02$. In order to satisfy Eq. (8), robot speeds are restricted to approximately 10 ± 1 cm/sec with positioning errors of $E_t = 2$ cm and orienting errors of $E_\theta = 5^\circ$. As we will see, only those motion primitives that result in a contact force less than $t_{\max} = 5$ N are allowed.

IV. Abstraction and Algorithms for Cooperative Planar Manipulation

In this section, we will focus on using the proposed abstraction framework in Section II to construct a simple kinematic sufficient abstraction for the original complex quasi-static model in Section III-B. Then, we will briefly describe the planning and tracking control algorithms using such abstraction.

A. Overview and purpose of abstraction

With the notation of Section II, the original complex quasi-static model in Section III-B is represented by $\mathcal{M}_i = \{f_i, X_i, U_i, \mathcal{P}_i, T_i\}$, where f_i is given in Equations (2-7), X_i includes all configurations of the robots and part, U_i includes all the inputs $\{v_i, \phi_i, n_i, c_i\}$ for robots, $\mathcal{P}_i$ includes all system parameters (part geometry, friction coefficients μ_s and μ_c, and three support points $\{x_i, y_i\}$), and T_i includes the time intervals of arbitrary length.

The objective of abstraction is to be able to predict the motion of the part via an efficient approximation of the reachable set. Since the original system has complicated motion behaviors under uncertainties in sensing, actuation, and system parameters, (*e.g.* unknown and changing three support points), it is impossible to compute a reachable set approximation for general inputs. Instead, we construct an abstraction model, which consists the motions of a finite set of *robust motion primitives* for which the following properties of interest are satisfied and preserved under uncertainties such that their reachable sets can be computed.

1) Two contacts between the robots and part, *i.e.*,.

$$\lambda_{n,j} > 0, j = 1, 2. \tag{10}$$

2) Straight line motion at speeds that imply quasi-static dynamics, (Eqs. 8 and 9).

In other words, for a trajectory $\tilde{x}_i : [0, t_f] \to X_i$, the property function Φ returns 1 if Eqs. 10, 8 and 9 are satisfied for any $t \in [0, t_f]$.

A robust motion primitive for the model $\mathcal{M}_i$ is a nominal control $\tilde{u}_i : [0, t_f] \to U_i$ with $\tilde{u}_i(0) = u_0$ and nominal $x_0 \in X_i$ such that with respect to nominal parameter history $\tilde{p}_i \in \mathcal{P}_i$ and $\tilde{p}_i(0) = p_0$

$$\Phi(\tilde{x}_i(x_0 + \delta x_i, \tilde{u}_i + \delta u_i, \tilde{p}_i + \delta \tilde{p}_i, t)) = 1 \tag{11}$$

for $t \in [0, t_f]$ and uncertainties δx_i in sensing, δu_i in actuation, and $\delta \tilde{p}_i$ in system parameters.

After constructing a finite set of robust motion primitives, the resulting sufficient abstraction model $\mathcal{M}_j = \{f_j, X_j, U_j, \mathcal{P}_j, T_j\}$ will be: f_j only include kinematic part of f_i. X_j is still the same as X_i under the identify map Θ_j^i. U_j is a discrete subset of U_i, each of which corresponds to a resulting motion of a robust motion primitive. T_j only includes a set of time intervals of specific length corresponding to each input in U_j.

Because the dynamics of $\mathcal{M}_j$ are invariant (or robust) to the three support points and friction coefficient, we are able to ignore the three point support parameters and friction coefficients in the system parameter set $\mathcal{P}_j$. This greatly reduces the uncertainty in the abstraction model.

While we will use the specific example and parameters described in Section III-E in our development, the same ideas are extensible to *any* planar object and to any set of position controlled robots.

B. Construction of robust motion primitives

The fundamental question in searching for robust motion primitives is to whether the properties of interest are preserved over continuous uncertainty sets. This is a very challenging verification problem for which the state-of-the-art does not include a solution for general systems [15]. Instead, motivated by recent work on sampling-based verification and falsification [10], we use the numerical Monte Carlo simulation method to construct robust motion primitives by checking whether the properties of interest are preserved with respect to a finite number of samples. This at least provides us with a computational tool for complex manipulation systems.

The construction process is carried out in the following steps:

1) Sample $\tilde{u}$ from U_i and x_0 from X_i.
2) Check whether $\Phi(\tilde{x}(x_0, \tilde{u}, \tilde{p}, 0)) = 1$.
   This step check whethers Φ is satisfied at the starting moment by solving the Mixed Complementarity Problem defined by Eqs. 2-7. Theoretically, there are no results on the uniqueness and existence of the solution for the quasi-static problem with multiple rigid bodies under two pushing contact. So, when there is no solution for a given manipulation, we simply say that Φ is not satisfied. When multiple solutions are observed, we say that Φ is preserved if it is satisfied in all solutions.
3) Check whether $\Phi(\tilde{x}(x_0 + \delta x, \tilde{u} + \delta u, \tilde{p} + \delta \tilde{p}, 0)) = 1$ for uncertainties δx, δu, and $\delta \tilde{p}$.
   This step checks whether the properties of interest are satisfied at the starting moment with respect to uncertainties in sensing, actuation, and system parameters. We consider the uncertainties in the three support points, robot velocity magnitude, and friction coefficients that are chosen from three bounded continuous sets. We first generate a finite set of three support point samples,

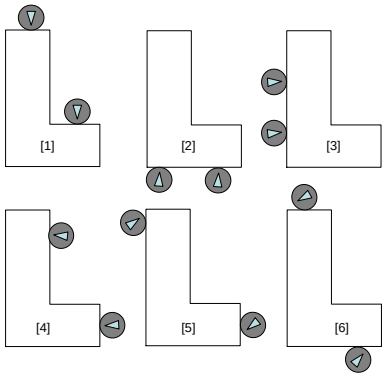

Fig. 5. Robust translational ([1]-[4]) and rotational ([5], [6]) motion primitives

velocity samples, and friction coefficient samples, and then use the procedure in (2) to check whether Φ is satisfied for all these samples. If true, the manipulation is identified as robust; otherwise, it is non robust.

4) Check whether $\Phi(\tilde{x}(x_0+\delta x, \tilde{u}+\delta u, \tilde{p}+\delta\tilde{p}, t)) = 1$ for uncertainties δx, δu, and $\delta\tilde{p}$ in at a finite set of discrete times $t \in [0, t_f]$.
This step checks whether Φ is preserved over the entire duration of a robust motion primitive. The time interval is a bounded continuous set.

Note that for a high dimensional input space, this search process is very computationally expensive. However, we only need such a computation once in the preprocessing. Alternatively, in this paper, we use a set of candidates for robust motion primitives from human intuition instead of random samples.

C. Constructed robust motion primitives

Robust motion primitives for an L-shape part are shown in Fig. 5 (see detailed parameters in Section V). We will now analyze the reachable sets of these primitives under uncertainties, which will help design of the tracking control along a given path.

Abstraction reduces the uncertainties due to the friction coefficient, intermittent contact, and unknown three support points. Uncertainties in sensing still exist and cause the part to vary from its nominal trajectory. However, the preserved properties in the abstraction enable us to predict the motions of the part under these uncertainties by estimating the bounds on their reachable set. In the following, we will compute bounds on these variations as a function of pushing distance d based on kinematic analysis of the abstraction model. These bounds will help to design planning and tracking control algorithms on top of the abstraction model.

1) Bounding the reachable set for the motion in Fig. 5[1-4]: The concept is illustrated with the example shown in Fig. 6. When the intermittent frictional contact modes switch from the left to the right, the maximal offset in y is generated by the following situation. We will assume that two robots push with the same nominal velocity v along the positive x direction with nominal separation distance d_s. Initially, the top and bottom robots respectively have rolling (**R**) and sliding (**S**) contacts (Fig. 6.1). When the coordination errors between two robots results in an error of E_d (Fig. 6.2), the contact modes switch to **S** (top) and **R** (bottom) (Fig. 6.3). Then, when the errors drive

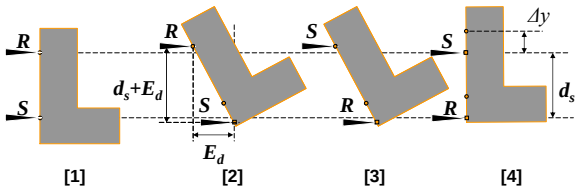

Fig. 6. The position offset in y caused by intermittent contact modes from the left to the right, in which **R** and **S** respectively stand for rolling and sliding.

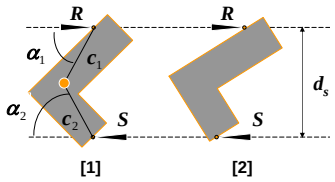

Fig. 7. Constant contact modes cause maximal changes of the configuration.

the robots back to their nominal separation, it can be observed that an offset in y has been generated (Fig. 6.4). Furthermore, when such switching pattern is executed multiple times, the offset in y will be accumulated.

More generally, if both contacts are not on the same edge, the top and bottom contact points are on edges respectively with orientation γ_1 and γ_2. The vector from the top contact point to the bottom has length l and angle χ. We have the following upper bounds over the whole pushing distance d,

$$y \in \begin{array}{l}[-\max(\Delta d_1 \sin\gamma_1, \Delta d_2 \sin\gamma_2)\lceil \frac{E_v d}{4E_d v}\rceil - E_t, \\ \max(\Delta d_1 \sin\gamma_1, \Delta d_2 \sin\gamma_2)\lceil \frac{E_v d}{4E_d v}\rceil + E_t].\end{array} \quad (12)$$

Similarly, the reachable sets of x and θ are respectively bounded by

$$x \in [-E_d + d, E_d + d], \quad (13)$$

$$\theta \in [-E_\theta - \tan^{-1}\frac{E_d}{d_s - E_d}, E_\theta + \tan^{-1}\frac{E_d}{d_s - E_d}]. \quad (14)$$

2) Bounding the reachable set for the motion in Fig. 5 [5-6]: The reachable set under this pushing is also bounded by analyzing the potential contact model switching patterns.

In Fig. 7, when the top and bottom contacts are respectively always rolling and sliding, x reaches its maximal value, which is less than $d + E_d$. Similarly, x is larger than $-d - E_d$ when the top and bottom contact modes are respectively sliding and rolling. Therefore, we have

$$x \in [-d - E_d, d + E_d], \quad (15)$$

$$\theta \in [-E_\theta, \tan^{-1}\frac{2d + d_h + E_d}{d_s - E_d} - \tan^{-1}\frac{d_h - E_d}{d_s + E_d} + E_\theta = \Delta\theta], \quad (16)$$

$$\begin{array}{l} y \in [-c_2(\sin(\alpha 2 + E_\theta) - \sin(\alpha 2 - \Delta\theta - E_\theta)) - E_t, \\ c_1(\sin(\alpha 1 + E_\theta) - \sin(\alpha 1 - \Delta\theta - E_\theta)) + E_t], \end{array} \quad (17)$$

in which d_h is the horizontal distance between two contact points at the starting moment of the manipulation in Fig. 7.1.

D. Planning and tracking control algorithms

For the L shape part in this paper, the four robust translation motion primitives and two robust rotational motion primitives

in Section IV-C are sufficient to achieve small time local controllability of the part.

For a general polygonal shape part, we need three translational robust motion primitives and two rotational motion primitives to achieve the small time locally controllable property. Any two of them should not be collinear and the dot product of three direction vectors should be negative. Two rotational motion primitives should be in opposite directions.

Given a path with a given tracking precision E_p, we are able to iteratively track the path using these motion primitives. In each iteration, we compute the pushing distance d for a given robust motion primitive with respect to the required tracking precision. The reachable set of the part after pushing distance d should be bounded inside the E_p-neighborhood of the nominal trajectory. After the pushing, if the part is off the nominal path, robust motion primitives are used to push the part back to the nominal trajectory. In this way, we are able to track any given path with precision up to the sensing and actuation limit.

With this tracking control algorithm, we solve the challenging cooperative manipulation problem by first using a sampling based path planning algorithm, *e.g.* PRM [16] or RRT [17], to compute a collision-free path and then execute by path tracking that relies on robust motion primitives.

V. Experimental Results

We have collected experimental results to demonstrate both the validity of robust motion primitives for mobile robot manipulation as well as the effectiveness of these primitives applied to manipulation/assembly tasks. First, we demonstrate that robust motions (due to the definition in Section IV-B) are feasible for our system then we go on to validate the reachable sets for the primitives derived in Section IV-C. Because we have a conservative estimate of the set of states that can be reached by applying a motion primitive, we can construct a tracking control system that can use robust motion primitives to follow an arbitrary path. Finally, we demonstrate that a simple planning algorithm in conjunction with these techniques can be used to complete a cooperative manipulation task.

A. System parameters for the experimental platform

All experiments are conducted on a multi-robot testbed [20] utilizing a team of small differential drive robots (radius 0.15m) and an overhead tracking system for localization of both the manipulated object and the robots. The position and orientation sensing error due to the tracking system are respectively $E_t = 2$cm and $E_\theta = 5^\circ$. Each robot is controlled using a feedback linearization scheme to tow along a desired trajectory but there is no feedback of relative state information. In other words, while each robot is controlling its own state to execute a straight line maintaining the abstraction in Section IV-A, the cooperative manipulation primitive is executed in an open loop fashion. This introduces additional error in their relative position control bounded by $E_d = 4$cm. Each robot is able to control its velocity within an error of $E_v = 1$ cm/s.

As mentioned earlier, the robots are manipulating an L-shape with a characteristic length of 1m, mass of 2.5Kg, and an approximate coefficient of friction with the floor of $\mu_s = 0.08$. Each robot has a mass of 8.6Kg and coefficient of friction with the L-shape of $\mu_c = 0.6$.

	$vert_1$	s_1	v_1	ϕ_1	$vert_2$	s_2	v_2	ϕ_2
+ X	5	0.9	0.1	0.0	5	0.1	0.1	0.0
+ Y	4	0.7	0.1	0.0	4	0.1	0.1	0.0
- X	1	0.1	0.1	0.0	3	0.3	0.1	0.0
- Y	0	0.1	0.1	0.0	2	0.3	0.1	0.0
+ θ	0	0.1	0.1	-0.7	4	0.1	0.1	-0.7
- θ	5	0.9	0.1	0.7	3	0.3	0.1	0.7

TABLE I

Details for our robust motion primitives, in which s_1 and s_2 are measured in m, ϕ_1 and ϕ_2 are in rad, and v_1 and v_2 are in m/s.

B. Non-robust motions

To highlight the advantage of using robust motion primitives, Fig. 8 depicts non-robust manipulations. Two-point contact is not maintained in these non-robust examples and the motion is unpredictable under the uncertainty inherent to the system. On the other hand, as we will now show, the result of robust motion primitives can be analytically bounded based on the assumptions of system uncertainty.

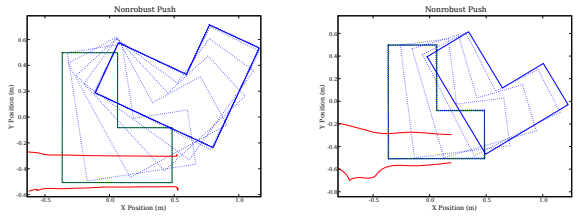

Fig. 8. Non-robust motion primitives.

C. Validation of reachable sets

To compute the bounds on the reachable sets due to motion primitives on the L-shape, we must evaluate the equations presented in Section IV-C. For translation, there is a nonlinear system of equations that must be solved numerically. Other than this step, the rest of the calculations are straight forward given the parameters of the primitive and the uncertainty of the system. Each motion primitive is parameterized as specified in Fig. 5 with the values in Table I. Additionally, the reachable set is a function of the pushing distance d.

We conducted several trials with different configurations of the two canonical manipulation primitives for translation and rotation to show that the resulting trajectories of the manipulated object always lie within the computed bounds. Fig. 9 depicts the part trajectories under different robust manipulation primitives to demonstrate that the final part positions lie within the analytically calculated bounds. Tables II and III provide details on the reachability sets for two sample motion primitives.

We have observed that in experiments, the motion primitives nearly always overshoot the upper bound along the direction of the desired motion. This is the result of assuming a kinematically controlled robot when there are, in fact, acceleration limits. However, it is a simple task to account for this with adequately enlarged reachability bounds or better low-level position control.

D. Validation of the tracking control

Since, for a given motion primitive and pushing distance d, we can calculate bounds the reachable set of the part, it

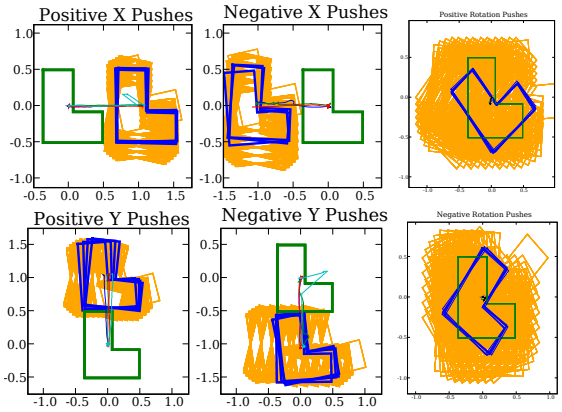

Fig. 9. Part trajectories respectively due to four configurations of the translation primitive ($\pm x, \pm y$) and two configurations of the rotation primitive ($\pm\theta$) overlaid on a sampling of the reachable set.

	Positive X				**Positive** Y		
	x(m)	y(m)	θ (°)		x(m)	y(m)	θ (°)
Predicted				*Predicted*			
Max	1.02	0.13	8.40	Max	0.19	1.02	8.67
Min	0.98	-0.13	-8.40	Min	-0.19	0.98	-8.67
Measured				*Measured*			
Avg	1.05	0.01	-0.11	Avg	0.01	1.05	-0.11
Max	1.06	0.02	1.72	Max	0.02	1.06	1.72
Min	1.04	-0.01	-1.32	Min	-0.01	1.04	-1.32

TABLE II

BOUNDS ON ROBUST TRANSLATION PRIMITIVE OVER FOUR TRIALS

is possible to design a tracking control algorithm that can efficiently follow a path within an E_p-neighborhood. Such a tracking controller can be thought of as a hybrid system that switches between modes for (1) correcting orientation to align the part tangent to path, (2) providing correction perpendicular to the path, and (3) pushing the part along a segment of the path. Each mode must ensure that d is chosen such that the reachable set after pushing will lie within E_p of the path while attempting to minimize tracking error or maximize distance traveled along the path. Thus the tracking controller will employ larger magnitude pushes along paths with larger E_p-neighborhoods.

Transitions between robust primitives are currently handled by a simple circular trajectory that each robot can follow to reach the initial conditions necessary to begin the next desired robust motion primitive. A less conservative but more general approach for transitioning between primitives in a complex environment is a challenging problem that we will address in future work.

	Positive θ				**Negative** θ		
	x(m)	y(m)	θ (°)		x(m)	y(m)	θ (°)
Predicted				*Predicted*			
Max	0.56	0.21	42.47	Max	0.27	0.49	4.58
Min	-0.58	-0.19	-4.58	Min	-0.31	-0.44	-47.50
Measured				*Measured*			
Avg	-0.02	-0.07	40.11	Avg	0.01	-0.01	-37.07
Max	-0.00	-0.06	43.14	Max	0.02	0.00	-34.38
Min	-0.03	-0.08	37.87	Min	-0.01	-0.03	-41.37

TABLE III

BOUNDS ON ROBUST ROTATION PRIMITIVE OVER SEVEN TRIALS

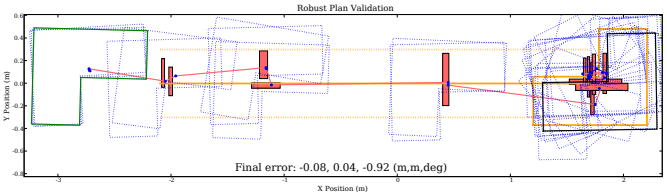

Fig. 10. Example of tracking control with robust motion primitives along a plan with small collision free zone. The bounding box for the reachability set of each translational push is shown.

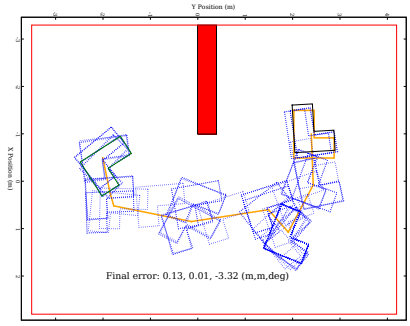

Fig. 11. Trajectory of L-Shape while being pushed along planned obstacle free path with robust motion primitives.

Fig. 10 depicts an example path with a relatively small E_p-neighborhood to show how the tracking controller must use a sequence of pushes and corrections to accurately follow the path.

E. Validation with a manipulation task

Finally, we solve the multiple robot manipulation problem in which two robots must cooperatively push a part from an initial to goal configuration though an environment with obstacles such as that shown in Fig. 1.

We use a sample-based algorithm, such as PRM or RRT, to generate a collision-free path from the initial configuration to the goal configuration which is then tracked with robust motion primitives. Snapshots of the experimental results are shown in Fig. 12, in which the solid piecewise-linear line connecting the initial and goal configurations is the collision free path from the path planning, the curves followed by the robots are the nominal controls to achieve robust motion primitives, and the wire-frame rectangular box represents a virtual obstacle. Fig. 11 shows the resultant trajectory of the L-shape during manipulation.

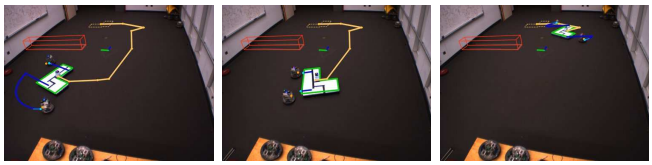

Fig. 12. Snapshots of cooperative manipulation of the L-shape part with two robots

VI. CONCLUSION

In this paper, we proposed a framework to develop abstractions for quasi-static manipulation tasks with uncertainty arising primarily from friction and unknown support points

and from errors in control and sensing. The abstractions were used to design algorithms for planar manipulation with cooperating mobile robots and the proposed approach was successfully validated with extensive experimental results. The manipulation system enables two autonomous robots to cooperatively push a part to a given goal configuration with a precision given by the errors in sensing and control.

There are several directions for ongoing work. First, our planning algorithm is very simple and generates very conservative paths. Clearly a better planner will achieve paths in more constrained environments. Because our focus was mainly on abstractions and control, we used a relatively simple planner. However, we are working on refining our planner for more cluttered environments. We are studying robust motion primitives with contact between the part and environment which will lead to a better abstraction for planning in the constrained space. We are also considering extension of the manipulation planning algorithm in [27, 31] to incorporate constraints from the movable part during the manipulation. Second, we used communication-less motion primitives with straight line robot trajectories in this work. Clearly, if robots can communicate, more complex trajectories can be generated and better performance can be obtained. However, this leads to more complexity in the formulation. We are currently studying if it is possible to derive more powerful abstractions that will exploit these additional capabilities.

The main conclusion of this work is simple. If the right abstractions can be derived for manipulation systems, powerful algorithms can also be derived to solve manipulation problems with uncertainties. Indeed, if we look at the examples in this paper and those in [9], it should be clear that the *same* planning algorithms can be used to solve manipulation problems across multiple length scales.

REFERENCES

[1] S. Akella, W. H. Huang, K. M. Lynch, and M. T. Mason. Parts feeding on a conveyor with a one joint robot. *Algorithmica*, 26(3/4):313–344, March/April 2000.

[2] S. Akella and M. Mason. Posing polygonal objects in the plane by pushing. In *Proceedings IEEE International Conference on Robotics & Automation*, pages 2255–2262, 1992.

[3] D. J. Armstrong. The quarks of object-oriented development. *Communications of the ACM*, 49(2):123–128, 2006.

[4] J. D. Bernheisel and K. M. Lynch. Stable pushing of assemblies. In *Proceedings IEEE International Conference on Robotics & Automation*, 2005.

[5] R.-P. Berretty, K. Goldberg, M. Overmars, and A. van der Stappen. Computing fence designs for orienting parts. *Computational geometry: theory and applications*, 10(4):249–262, 1998.

[6] A. Bicchi and V. Kumar. Robotic grasping and manipulation. In N. Siciliano, Bicchi and Valigi, editors, *Ramsete: Articuated and Mobile Robots for Services and Technology*, volume 270 of *Lecture Notes in Control and Information Sciences*. Springer Verlag, 2001.

[7] S. Blind, C. McCullough, S. Akella, and J. Ponce. Manipulating parts with an array of pins: A method and a machine. *International Journal of Robotics Research*, 20(10):808–818, Dec. 2001.

[8] M. Buehler, D. E. Koditschek, and P. J. Kindlmann. Planning and control of robotic juggling and catching tasks. *International Journal of Robotics Research*, 13(2):101–118, 1994.

[9] P. Cheng, D. Cappelleri, B. Gavrea, and V. Kumar. Planning and control of meso-scale manipulation tasks with uncertainties. In *Robotics: Science and Systems*, 2007.

[10] P. Cheng and V. Kumar. Sampling-based falsification and verification of controllers for continuous dynamic systems. In S. Akella, N. Amato, W. Huang, and B. Misha, editors, *Workshop on Algorithmic Foundations of Robotics VII*, 2006.

[11] S. Collins, A. Ruina, R. Tedrake, and M. Wisse. Efficient bipedal robots based on passive dynamic walkers. *Science Magazine*, 307:1082–1085, 2005.

[12] M. R. Cutkosky. *Robotic Grasping and Fine Manipulation*. Kluwer Academic Publishers, Norwell, Massachusetts, 1985.

[13] J. Fink, N. Michael, and V. Kumar. Composition of vector fields for multi-robot manipulation via caging. In *Robotics: Science and Systems Conference*, 2007.

[14] O. C. Goemans, K. Goldberg, and A. F. v. d. Stappen. Blades for feeding 3d parts on vibratory tracks. *Assembly Automation Journal*, 26, 2006.

[15] T. A. Henzinger, P. W. Kopke, A. Puri, and P. Varaiya. Whats decidable about hybrid automata? *J. Comput. Syst. Sci.*, 57:94–124, 1998.

[16] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics & Automation*, 12(4):566–580, June 1996.

[17] S. M. LaValle and J. J. Kuffner. Rapidly-exploring random trees: Progress and prospects. In *Proceedings Workshop on the Algorithmic Foundations of Robotics*, 2000.

[18] K. M. Lynch and M. T. Mason. Stable pushing: Mechanics, controllability, and planning. *International Journal of Robotics Research*, 15(6):533–556, 1996.

[19] M. T. Mason. Mechanics and planning of manipulator pushing operations. *International Journal of Robotics Research*, 5(3):53–71, 1986.

[20] N. Michael, J. Fink, and V. Kumar. Experimental testbed for large multi-robot teams: Verification and validation. *IEEE Robotics and Automation Magazine*, Mar. 2008.

[21] G. A. S. Pereira, V. Kumar, and M. F. M. Campos. Decentralized algorithms for multirobot manipulation via caging. *International Journal of Robotics Research*, 2004.

[22] M. Peshkin and A. Sanderson. The motion of a pushed sliding workpiece. *International Journal of robotics and automation*, 4:569–598, 1988.

[23] J. Ponce and B. Faverjon. On computing three-finger force-closure grasps of polygonal objects. *IEEE Transactions on Robotics & Automation*, 11(6):868–881, 1995.

[24] J. Ponce, S. Sullivan, A. Sudsang, J.-D. Boissonnat, and J.-P. Merlet. On computing four-finger equilibrium and force-closure grasps of polyhedral objects. *International Journal of Robotics Research*, 16(1):11–35, Feb. 1997.

[25] M. Raibert. *Legged Robots That Balance*. The MIT Press, 1986.

[26] J. K. Salisbury and J. J. Craig. Articulated hands: Force control and kinematic issues. *International Journal of Robotics Research*, 1(1):4–17, Spring 1982.

[27] T. Siméon, J.-P. L. J. Cortés, and A. Sahbani. Manipulation planning with probabilistic roadmaps. *International Journal of Robotics Research*, 23(1-8), 2004.

[28] P. Song, J. Pang, and V. Kumar. A semi-implicit time-stepping model for frictional compliant contact problems. *International Journal for Numerical Methods in Engineering*, 60:2231–2261, 2004.

[29] P. Song, J. Trinkle, V. Kumar, and J.-S. Pang. Design of part feeding and assembly processes with dynamics. In *Proceedings IEEE International Conference on Robotics & Automation*, Apr. 2004.

[30] D. Stewart and J. Trinkle. An implicit time-stepping scheme for rigid body dynamics with inelastic collisions and coulomb friction. *International Journal of Numerical Methods in Engineering*, 39:2673–2691, 1996.

[31] M. Stilman and J. Kuffner. Planning among movable obstacles with artificial constraints. In *Proceedings Workshop on Algorithmic Foundations of Robotics*, 2006.

[32] A. Sudsang, J. Ponce, and N. Srinivasa. Grasping and in-hand manipulation: Geometry and algorithms. *Algorithmica*, 26:466–493, 2000.

[33] A. Sudsang, F. Rothganger, and J. Ponce. Motion planning for disc-shaped robots pushing a polygonal object in the plane. *IEEE Transactions on Robotics & Automation*, 18(4):550–562, 2002.

[34] J. Trinkle, S. Berard, and J. Pang. A time-stepping scheme for quasistatic multibody systems. In *Proceedings, ASME International Design Engineering Technical Conferences*, September 2005.

[35] N. Zumel and M. Erdmann. Nonprehensible two palm manipulation with non-equilbrium transitions between stable states. In *Proceedings IEEE International Conference on Robotics & Automation*, pages 3317–3323, Apr. 1996.