Local Trajectory Stabilization for Dexterous Manipulation via Piecewise Affine Approximations

Weiqiao Han and Russ Tedrake

***Abstract*— We propose a model-based approach to design feedback policies for dexterous robotic manipulation. The manipulation problem is formulated as reaching the target region from an initial state for some non-smooth nonlinear system. First, we use trajectory optimization to find a feasible trajectory. Next, we characterize the local multi-contact dynamics around the trajectory as a piecewise affine system, and build a funnel around the linearization of the nominal trajectory using polytopes. We prove that the feedback controller at the vicinity of the linearization is guaranteed to drive the nonlinear system to the target region. During online execution, we solve linear programs to track the system trajectory. We validate the algorithm on hardware, showing that even under large external disturbances, the controller is able to accomplish the task.**

I. Introduction

How to enable robots to manipulate objects dexterously like human hands do is a long-standing problem [1]. Alongside hardware, perception, and planning, motor control is one of the challenges in manipulation. Designing reliable feedback controllers for manipulation is hard, due to the nonlinear and contact-rich nature of the manipulation tasks. For example, how should we design a controller for the robot to flip a carrot (half-cylinder) with the flat surface facing upwards (Fig. 1) to the pose where the flat surface is facing downwards? This is more than a simple pick-and-place task and indeed even an experienced human operator tele-operating the robot often cannot succeed in one or two tries (see the accompanying video).

In general, there are two main categories of approaches to control design for manipulation – model-based approaches and learning-based approaches. Model-based approaches have mainly been applied to grasping [2], [3], and planar pushing [4], [5]. They are usually specific to the hardware. Most model-based controllers are open loop [6]. In order to achieve the dexterity of a human hand, we want the robot to do tasks more complicated than grasping and planar pushing in a feedback fashion.

On the other end of the spectrum, learning-based approaches have been applied to tasks with greater variety and difficulty, ranging from moving the end-effector to a target pose [7], grasping [8], and planar pushing [9], to rotating a long rod [10], throwing objects [11], and rotating a cube [12]. Though learning-based approaches have been successful on tasks that model-based approaches have not been able to solve, they lack reliability or stability guarantees. Motions of manipulators are unpredictable, especially for those whose

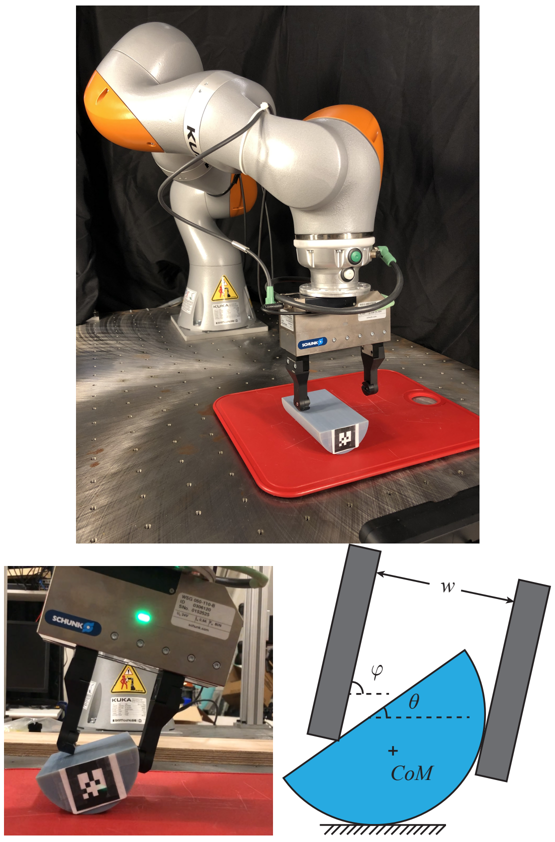

Fig. 1: Flipping a carrot (half-cylinder) using a parallel gripper

policies are represented by deep neural networks. Judging the usefulness of such control policies is always empirical.

Our approach falls into the first category. We present a general algorithm of feedback design for dexterous manipulation. We draw ideas from recent advances in humanoid robots path planning [13] and robust control synthesis [14], [15]. We formulate the manipulation problem as using a manipulator to change the object pose from its initial pose to a target pose. Our algorithm consists of the following steps. First, the pose of the object and that of the robot are incorporated into the system state, and a trajectory optimization method that has been used to design nominal trajectories for humanoid robots walking over uneven terrains is deployed to design a nominal trajectory for manipulating the object. Second, piecewise-affine (PWA) linearization around the nominal trajectory is computed and a funnel around the nominal mode of the PWA linearization is formed.

Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, 77 Massachusetts Avenue, Cambridge, MA 02139, USA. `weiqiaoh,russt@mit.edu`

Third, a linear program (LP) is solved at each time step during online execution, achieving real time feedback control. The algorithm can be implemented on common robotic platforms. It can be applied to more complicated tasks than grasping and planar pushing, and in particular, it can flip the half-cylinder. The controller is robust to moderate external disturbances. Our contributions are (1) designing a feedback control algorithm for dexterous manipulation, bridging the gap between locomotion/UAV control and robotic manipulation; (2) providing robustness guarantees for our algorithm (Proposition 1); (3) validating the algorithm on hardware.

II. Related Work

Model-based feedback control for manipulation: Lynch's group used feedback control for sliding [16], rolling [17], vibratory manipulation [18], [19], and hybrid manipulation using motion primitives [6], [20]. They designed specific manipulators for specific tasks. In contrast, our algorithm is more general and can be carried out on common robot platforms. Rodriguez's group used feedback control for planar pushing [21]. They linearize the nonlinear system and solve linear model predictive control (MPC) online to plan the trajectory.

Path planning: There are generally two categories of approaches to path planning. One is motion planning algorithms [22]. For discretized configuration space, grid search algorithms such as A* and its variants [23] are widely used. Sampling-based algorithms such as rapidly exploring random trees [24] are common approaches for continuous configuration space. The other category is the trajectory optimization algorithms. A common approach in this category would be to formulate the problem as nonlinear optimization programs and solve it using off-the-shelf numerical solvers [25]–[27]. Other approaches in this category include augmented Lagrangian [28], mixed-integer convex optimization (MICP) [29], differential dynamic programming (DDP) [30], and iterative linear quadratic Gaussian (iLQG) [31]. A combination of these two methods have proved even more useful [32], [33]. In this work, we use trajectory optimization and use off-the-shelf numerical solvers to solve a nonlinear optimization program.

Local feedback controllers: In control literature, it is quite standard to track a system trajectory using linear quadratic servo (LQ servo) or time-varying linear quadratic regulator (TVLQR) based on linearization of the system trajectory around nominal states [34]. In reinforcement learning, it is common to learn local linear models of the system and linear feedback control gains [7], [10]. However, the local linear model does not fully capture the contact-rich nature of manipulation. Robot fingers may make and break multiple contacts with the object due to small disturbances. When contact modes change, the dynamics may change. So it is more natural to model the local dynamics as a PWA system, i.e., a system whose state-input space is partitioned into several polytopic regions, with each region associated with a different affine dynamics equation.

However, stabilizing PWA systems alone is not an easy problem. Explicit solutions of optimal control for PWA systems can be computed offline by multi-parametric programming [35]–[42]. The computational complexity of these methods grows exponentially with respect to the number of time steps. Lyapunov-based approaches [43]–[45] and occupation measure approaches [46], [47] do not depend on the number of time steps, but are quite conservative and may not always find solutions. Sampling-based methods [15], [48] suffer from the issue of scalability. In this work, we consider manipulation problems in which there is only one rigid object, the manipulators are fully actuated, and the PWA dynamics is caused by manipulators making and breaking contacts with the object. We track the system trajectory by solving LP online.

III. Problem Statement and Approach

In many manipulation problems, the goal is for the manipulator to change the pose of an object from the initial pose to some target pose(s) specified by the user, for example, using a parallel gripper to flip a half-cylinder from the pose with the flat surface facing upwards to the pose with the flat surface facing downwards. We incorporate the pose of the manipulator and the pose of the object into the system state x. Then the manipulation problem is turned into a control problem: Design a feedback controller $u(x)$ that drives the initial state x_0 to a target region X_G. The dynamics of the manipulator-object system, $\dot{x} = f(x, u)$, is nonlinear and non-smooth.

Our algorithm contains both offline planning and online execution phases. During the offline planning phase, we formulate a nonlinear trajectory optimization problem that drives x_0 to a target state $x_N \in X_G$ in N time steps. We solve the trajectory optimization using off-the-shelf numerical solvers. The solution is a nominal trajectory $\{\bar{x}_0, \bar{u}_0, \ldots, \bar{x}_{N-1}, \bar{u}_{N-1}, \bar{x}_N\}$. This is an open-loop trajectory and might be fragile under external disturbances. We build funnels around (the linearization of) the nominal trajectory using polytopes. If the system state falls into the funnel, the system is guaranteed to reach the target region. In order to resist larger disturbances, we compute the PWA linearization around the nominal trajectory when the manipulator makes and breaks contacts with the object. As mentioned before, we assume that there is only one rigid object, that the manipulators are fully actuated, and that the local PWA dynamics is only caused by manipulators making and breaking contacts with the object. During online execution, we use a linear program to steer the system into the polytopes or onto the nominal points.

In summary, our algorithm consists of the following three steps: (1) solving nonlinear trajectory optimization to find the nominal trajectory offline; (2) computing the PWA linearization and building a polytopic funnel around the linearization of the nominal trajectory offline; (3) solving LP's online to drive the system around the nominal points.

IV. Trajectory Planning

A. *Trajectory Optimization*

We plan the path using trajectory optimization methods. In particular, we use direct transcription as in [26]. The continuous-time dynamics $\dot{x} = f(x,u)$ is discretized into a discrete-time system $x^+ = \phi(x,u)$ with sampling time dt. The trajectory is discretized into N time steps with $N \cdot dt = T$, where T is the time horizon:

$$\begin{aligned}
\text{minimize } & \sum_{t=0}^{T} L(x[t], u[t]) \\
\text{subject to } & m\ddot{r}[t] = mg + \sum_j F_j[t] \\
& I\ddot{\theta}[t] = \sum_j (c_j[k] - r[k]) \times F_j[t] \\
& \text{friction cone constraints, contact constraints,} \\
& \text{kinematics constraints, time integration constraints}
\end{aligned}$$

where $r[t]$ is the position of the center of mass, $F_j[t]$ are forces, $c_j[t]$ is the the contact position of the j-th force for each j, L is the loss function, and the decision variables include states $x[t]$, controls $u[t]$, and variables in the contact constraints. The variables $r[t]$ and $c_j[t]$ are part of the states $x[t]$, and the forces $F_j[t]$ are part of the controls $u[t]$.

The first two constraints are Newton-Euler equations. The friction cone constraints for planar systems are $-\mu F_{j,n} \leq F_{j,t} \leq \mu F_{j,n}$, where $F_{j,n}$ is the normal force and $F_{j,t}$ is the frictional force. For 3-dimensional systems, we can use a polyhedral cone to approximate the friction cone [49]: $F_j[t] = \sum_i \beta_{ij} w_{ij}, \beta_{ij} \geq 0$, where w_{ij}'s are the spanning vectors of the polyhedral cone. For some contacts, we formulate the contact constraints as linear complementarity problems (LCP) [25] to fully characterize all possible contact modes – sticking, sliding, or breaking contacts. We use IPOPT [50] to solve the trajectory optimization offline. Since IPOPT is an interior point method solver, the LCP constraints $P(x)^\top Q(x) = 0, P(x) \geq 0, Q(x) \geq 0$, can be replaced by the equivalent constraints $P(x)^\top Q(x) \leq 0, P(x) \geq 0, Q(x) \geq 0$, and further be relaxed as $P(x)^\top Q(x) \leq \epsilon, P(x) \geq 0, Q(x) \geq 0$ for small $\epsilon > 0$.

B. *Force as Control Input*

In the trajectory planning described in the previous subsection, we borrowed the idea of zero-moment point (ZMP) for bipedal footstep planning in the humanoid robot literature [51], [52]. For a bipedal robot walking on the ground, ZMP is by definition a point on the ground where the sum of all the tangential moments equals zero. Although many humanoid robots have pressure sensors on the feet, because of lack of reliability (in the case of Atlas), researchers do not measure ZMP directly. What they do is to plan a joint ZMP and center of mass (CoM) trajectory, and then only track the CoM trajectory during online execution [13].

Similarly, in the trajectory optimization formulation, we use forces as part of the control input. In the manipulation context, the forces are those between the gripper and the object, and those between the object and the environment, e.g. the table. Although the hardware we are using cannot directly measure the force it applies to the object, we still use forces as control variables to help plan the CoM trajectory of the object as well as the pose trajectory of the gripper. During online execution, we only track the trajectory of the CoM of the object and that of the gripper. We find in practice this approach works well.

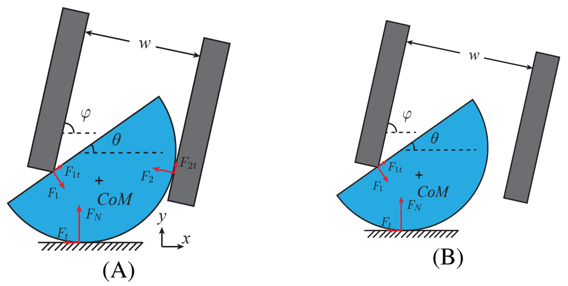

Fig. 2: Two contact modes. (A) Right finger in contact with carrot. (B) Right finger not in contact with carrot.

V. Local Feedback Control

A. *Local Multi-Contact Dynamics*

From the nonlinear trajectory optimization, we obtain a nominal trajectory $\{\bar{x}_0, \bar{u}_0, \bar{x}_1, \bar{u}_1, \ldots, \bar{x}_N\}$, where $\bar{x}_N \in X_G$. At each nominal point $(\bar{x}_i, \bar{u}_i)$, where $i = 0, \ldots, N$, we linearize the dynamics $\dot{x} = f(x,u)$ as $\dot{x} = A_i(x - \bar{x}_i) + B_i(u - \bar{u}_i) + c_i$, where $A_i = \frac{\partial f}{\partial x}(\bar{x}_i, \bar{u}_i), B_i = \frac{\partial f}{\partial u}(\bar{x}_i, \bar{u}_i)$, and $c_i = f(\bar{x}_i, \bar{u}_i)$. This continuous time affine system can be discretized as $x^+ = \tilde{A}_i x + \tilde{B}_i u + \tilde{c}_i$. This can equivalently be obtained by linearizing the discretized system $x^+ = \phi(x,u)$. The corresponding state space X_i^1 is obtained by linearizing all constraints $g(x,u) \leq 0$ at $(\bar{x}_i, \bar{u}_i)$.

Now we consider different contact modes due to making or breaking contacts between an object and the manipulator. We fix a nominal point $(\bar{x}_i, \bar{u}_i)$. Suppose there are $p \in \mathbb{N}$ contact locations that may make or break contacts. Each contact mode corresponds to a distinct dynamics $\dot{x} = f_j(x,u)$ and constraints $g_j(x,u) \leq 0$, where $j = 1, \ldots, s := 2^p$, $f_1 = f$, and $g_1 = g$. For the half-cylinder example, the right finger can be touching or not touching the half-cylinder, giving two contact modes with distinct dynamics and distinct state space regions (Figure 2). We call the contact mode in which the nominal trajectory is computed the *nominal mode* or Mode 1. We linearize the dynamics and the constraints for modes other than the nominal mode and evaluate at $(\bar{x}_i, \bar{u}_i)$. Since making and breaking contacts can happen when the system state makes very small changes, the linearization is in the vicinity of the nominal point and hence is valid. In fact, the nominal points can be on the boundaries of state space cells of a few modes. Thus we obtain a piecewise affine system $x^+ = A_{i,j}x + B_{i,j}u + c_{i,j} =: h_{i,j}(x,u)$ with state space X_i^j, $j = 1, \ldots, s$, around each nominal point $(\bar{x}_i, \bar{u}_i)$, $i = 1, \ldots, N$. We call $x^+ = h_{i,1}(x,u)$ the *nominal linearization*.

B. *Polytopic Funnel around Nominal Trajectory*

After we get a nominal trajectory, we are going to build a funnel around the trajectory so that if the system state is inside the funnel, it will always stay inside the funnel until reaching the target region. Funnels can be sum-of-squares (SOS) [14], [53] or polytopic [15]. We use the latter, because numerical computations involving polytopes requires solving LP or quadratic program (QP), which are more reliable than solving SOS programs.

Here we briefly review the polytopic tree method in [15]. Suppose the system is a time-varying affine system

$$x_{t+1} = A_t x_t + B_t u_t + c_t.$$

Given a target region X_G and time horizon N, the method computes a trajectory $\{\bar{x}_i, \bar{u}_i\}_{i=0}^N$ alongside with polytopes

$$Y_i = \{\bar{x}_i\} \oplus G_i \mathbb{P} \tag{1}$$

and a control law

$$u_i(x) = \bar{u}_i + \theta_i p(x) \tag{2}$$

around each point $(\bar{x}_i, \bar{u}_i)$ on the trajectory by solving a main LP. Here $\oplus$ represents Minkowski sum, G_i and θ_i are decision matrices the main LP searches over, $G_{i+1} = A_i G_i + B_i \theta_i$ captures the evolution of the polytopes over time with respect to the system dynamics, $\mathbb{P}$ is the hyper-cube $[-1, 1]^n$, and $p(x) \in \mathbb{P}$ satisfies

$$x = \bar{x}_i + G_i p(x). \tag{3}$$

The main LP to be solved offline encodes the state space constraints in polytopic containment forms, including the final polytopic containment constraint $Y_N \subseteq X_G$, as well as trying to maximize the volumes of the polytopes. During online execution, if the current state x is in some polytope Y_i, then $p(x)$ can be found by solving an LP through Equation (3) and $u_i(x)$ can be calculated by Equation (2). By following the control law $u_i(x)$, the system is guaranteed to land inside the next polytope Y_{i+1} and hence eventually it will reach $Y_N \subseteq X_G$.

We use the method to build polytopes for the nominal linearization $x^+ = h_{i,1}(x, u)$ around the nominal trajectory $(\bar{x}_i, \bar{u}_i)$. While [15] deals with PWA systems, we show that the polytopic tree method can be extended to non-smooth nonlinear systems.

Proposition 1. If $x^+ = \phi(x, u)$ and ϕ is Lipschitz, and if the polytopic tree method finds the polytopes Y_i as in Equation 1 for the nominal linearization $x^+ = h_{i,1}(x, u)$ at the nominal trajectory, with $\mathbb{P} = [-1, 1]^n$ and G_i full rank $\forall i$, then there exists $\mathbb{P}_i = [-a_i, a_i]^n, 0 < a_0 \leq a_1 \leq \dots \leq a_N = 1$ such that if $x \in \tilde{Y}_i := \{\bar{x}_i\} \oplus G_i \mathbb{P}_i$, then by following u as in Equation 2, $\phi(x, u) \in \tilde{Y}_{i+1}$. So x will eventually land in $\tilde{Y}_N = Y_N \subseteq X_G$.

Proof Sketch. If the system state $x \in \tilde{Y}_{N-1} = \{\bar{x}_{N-1}\} \oplus G_{N-1}\mathbb{P}_{N-1}$, by following u_{N-1}, $x^+ \in \{\bar{x}_N + e_{N-1}\} \oplus G_N \mathbb{P}_{N-1}$, where e_{N-1} is the residual error induced by the linearization. Since the dynamics ϕ is Lipschitz, e_{N-1} goes to 0 as (x, u) goes to $(\bar{x}_{N-1}, \bar{u}_{N-1})$. Given small $\varepsilon > 0$, we can find $\delta > 0$ such that if $||(x, u) - (\bar{x}_{N-1}, \bar{u}_{N-1})||_2 < \delta$, then $||e_{N-1}||_2 < \varepsilon$. We can choose ε and a_{N-1} such that any (x, u) satisfying $x \in \tilde{Y}_{N-1}$ and Equation 2 also satisfies $||(x, u) - (\bar{x}_{N-1}, \bar{u}_{N-1})||_2 < \delta$ and such that $e_{N-1} \oplus G_N \mathbb{P}_{N-1} \subseteq G_N \mathbb{P}_N$. Then $x^+ \in \tilde{Y}_N$. The proof is complete by repeat the procedure for $i = N-2, \dots, 0$ in this order. □

The Proposition says that if the system state is close enough to the nominal trajectory and if the nonlinear dynamics is Lipschitz, then we can use the linear control law (Equation 2) for the nominal linearization of the trajectory to steer the nonlinear system to the target region for sure. This gives us certain guarantees locally. We want to be able to handle larger external disturbances during online execution. This is what we are going to discuss in the next section.

VI. Online Execution

The polytopic tree method in [15] hopes to probabilistically cover the state space by polytopes by growing a single polytopic trajectory to an existing polytopic tree, similar in methodology to the growth of the LQR-trees [53]. The method samples points in the state space, steers sample points to the current polytopic tree as well as building polytopes along the way by solving mixed-integer linear program (MILP), and enlarges the current tree by adding the new polytopes to the tree.

In practice, for example for the half-cylinder flipping experiment, there are several problems with the polytopic tree method. First, the volumes of the polytopes are very small, and hence a state almost never falls into a polytope. Second, the polytopic tree method requires checking the closest polytope online, which is computationally inefficient in the naive implementation when there are large number of polytopes. Third, the polytopic tree method deals with PWA systems, but our system is nonlinear and computing trajectory from a sample point to the current tree is potentially an expensive nonlinear trajectory optimization problem which cannot be carried out online.

Therefore, we propose the practical improvement of the polytopic tree method, with the sacrifice of stability guarantees when there are large deviations to the nominal trajectory. We only keep one nominal trajectory, which is computed in Section IV. We build polytopes $Y_i = \{\bar{x}_i\} \oplus G_i \mathbb{P}, i = 0, \dots, N$ around the nominal linearization of the trajectory. This amounts to solving an LP. During online execution, we compute the closest nominal state $\bar{x}_i$ to the current state x (with respect to some weighted L_2 norm) and determine the current contact mode j. If x is inside the polytope $\{\bar{x}_i\} \oplus G_i \mathbb{P}$, the we use the corresponding control law $u_i(x) = \bar{u}_i + \theta_i p(x)$, where $x = \bar{x}_i + G_i p(x)$. Otherwise we let the target index be $v = \min\{i + 1, N\}$ and solve the following LP to get control u:

$$\begin{aligned} \min_{\gamma, p, \delta, u} \quad & \alpha^\top \gamma \\ \text{subject to } \quad & x_v + G_v p = h_{i,j}(x, u) + \delta \\ & p \in \mathbb{P}, |\delta_k| \leq \gamma_k, k = 1, \dots, n \end{aligned} \tag{4}$$

where α is some weight or cost vector. This LP means we want the state to get to the polytope with index v as close as possible. When, for example in the half-cylinder flipping experiment, the volumes of the polytopes are very small, we can directly solve the LP

$$\begin{aligned} \min_{\gamma,\delta,u} \quad & \alpha^\top \gamma \\ \text{subject to } \quad & x_v = h_{i,j}(x,u) + \delta \\ & |\delta_k| \le \gamma_k, k = 1, \dots, n \end{aligned} \tag{5}$$

which means we want the state to get to the nominal state with index v as close as possible.

Algorithm 1 Stabilizing controller around nominal trajectory

```
Input Current state $x \notin X_G$
Output Control $u$
1: if $x \in \{\bar{x}_i\} \oplus G_i \mathbb{P}$ for $i \in I$ then return $u = \bar{u}_i + \theta_i p(x)$,
   where $x = \bar{x}_i + G_i p(x)$, $i$ is the largest element in $I$.
2: Find the closest nominal state $\bar{x}_i$ to $x$, w.r.t. some
   weighted $L_2$ norm.
3: Determine the current contact mode $j$.
4: Let the target index be $v = \min\{i + 1, N\}$.
5: Solve LP (4) or (5), return $u$.
```

The procedure is summarized in Algorithm 1. There can be many variants to Steps 4 and 5. For example, one might use MPC-style planning based on local PWA linearization. During offline phase, one samples states $\tilde{x}_k$ not in the nominal mode and solve MICP to get to some target points $\bar{x}_{v(k)}$ on the nominal trajectory, hence storing a list of samples $\{(\text{state } \tilde{x}_k, \text{ mode sequence to get to target } \bar{x}_{v(k)})\}_{k=1}^{M}$. The target index $v(k)$ for each sample state $\tilde{x}_k$ can be chosen by comparing the cost to get to all nominal states $\bar{x}_i, i = 0, \dots, N$. During online execution, one finds the closest sample $\tilde{x}_k$ to the current state x and solve QP or LP to get to $\bar{x}_{v(k)}$ fixing the mode sequence as stored.

We find empirically that for the half-cylinder flipping experiment, solving LP like (4) or (5) to directly go to the nominal trajectory is more efficient than MPC-style planning which plans multiple steps to reach the nominal trajectory. This might be because of our assumptions that the change of contact modes is only caused by the manipulator making and breaking contacts with the object and that the manipulator is fully actuated. So instructing the manipulator to directly go back to the desired position works. Also MPC-style planning on linearized PWA systems may accumulate linearization errors.

During the online execution, we use $\mathbb{P} = [-1, 1]^n$ instead from $\mathbb{P}_i$ in Proposition 1, because it is more computationally efficient to use $[-1, 1]^n$ and it is hard to compute $\mathbb{P}_i$. Since $\mathbb{P}_i \subseteq \mathbb{P}$, we know once x happens to fall into $\{\bar{x}_i\} \oplus G_i \mathbb{P}_i$, then the system is guaranteed to reach the target region.

VII. Experiment

We carried out the experiment on a Kuka robot with a two-finger Schunk gripper. The half-cylinder representing the carrot is 0.11 m long and its radius is 0.036 m. We put an AR-tag [54] with side width 3 cm on a side of the half-cylinder and use a Kinect to track the pose of the half-cylinder. We can get in real time the pose of the gripper relative to the Kuka base and hence the pose of the gripper relative to the table. Once we compute the initial pose of the half-cylinder relative to the table, we can track the pose of the half-cylinder relative to the gripper in real time.

The goal is to flip the half-cylinder 180 degrees, i.e., to manipulate the half-cylinder with the flat surface facing upwards to the pose where the flat surface is facing downwards to the table. We use our algorithm to design a trajectory that flips the half-cylinder 90 degrees so that the grippers are holding the half-cylinder. After that, a manually-designed (open-loop) controller would transport the half-cylinder and flip another 90 degrees. We mainly focus on the first 90-degree rotation for several reasons. First, it is most challenging in the entire manipulation process, and manually designed open-loop controllers usually fail in this phase. Even an experienced human operator tele-operating the robot cannot accomplish this task in one or two attempts and the failures are often in the first 90-degree phase (see the accompanying video). Second, in the next 90-degree rotation, the dynamics is different from that in the first 90-degree rotation, so one needs to design a different trajectory separately, instead of simply extending the first trajectory. Besides, it is very simple to design a good open-loop controller for the next 90-degree rotation (see the accompanying video).

The state is $\mathbf{x} = [x, y, \theta, \dot{x}, \dot{y}, \dot{\theta}, \varphi, w]$, where x and y are the coordinates of CoM of the half-cylinder, θ is the angle between the flat surface of the half-cylinder and the horizontal axis, $\dot{x}, \dot{y}, \dot{\theta}$ are the first-order time derivatives of x, y, θ, respectively, φ is the angle between the finger of the gripper and the horizontal axis, and w is the separation between two fingers. The control input is $\mathbf{u} = [F_N, F_t, F_1, F_{1t}, F_2, F_{2t}, \dot{\varphi}, \dot{w}]$, where F_N, F_t are the normal force and the friction between the half-cylinder and the ground, and similar definitions for F_1, F_{1t}, F_2, F_{2t} (Figure 2). We set up trajectory optimization with time horizon $N = 100$ and sampling time $dt = 0.01$ s. We constrain that the contact point between the left finger and the half-cylinder does not change over the entire trajectory. The goal state region of the trajectory optimization is $X_G = \{\mathbf{x} : \varphi = \theta = 90°\}$. It took IPOPT about 2 seconds to solve the trajectory optimization on an Intel i7 3.3 GHz, 32 GB RAM machine[1].

We design controllers in 2D, and in execution the half-cylinder is 3D, so we always operate at the center section of the half-cylinder. During execution, we let the goal region be $\widetilde{X}_G = \{\mathbf{x} : \varphi = \theta, w \le c\} \supseteq X_G$ for some constant c. Once the state is in the goal region, the gripper is in grasp of the half-cylinder, ready for the remaining operations. The open-loop controller obtained from trajectory optimization

[1]By varying the time horizon or other constraints, the solving time of IPOPT varies from 1 to 25 seconds or IPOPT cannot find a feasible solution. We found that on solving this particular problem with varying constraints, IPOPT generally behaves better than SNOPT [55].

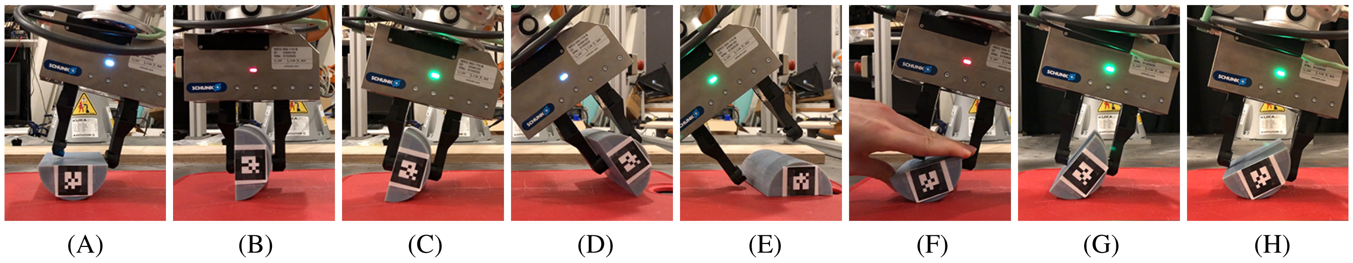

Fig. 3: Flipping the half-cylinder

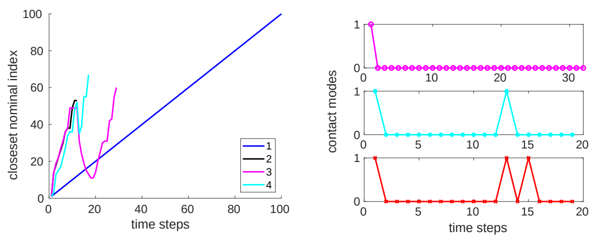

Fig. 4: The figure on the left plots the index of the closest nominal trajectory state for the current state *before* reaching the goal region $\widetilde{X}_G$ vs. time steps for four typical trajectories. Trajectory 1 can be thought of as the open loop trajectory and is just for reference. Trajectory 2, 3, and 4 are typical trajectories for the closed-loop controller, the closed-loop controller under the first type of disturbance, and the closed-loop controller under the second type of disturbance, respectively. They terminate early, because $\widetilde{X}_G$ is strictly larger than X_G. The figure on the right plots the contact modes for 3 typical trajectories. From top to bottom are typical contact modes for the closed-loop controller under the first type of disturbance, the closed-loop controller under the second type of disturbance, and the close-loop controller under the second type of disturbance where the gripper opens 4 cm.

Disturbance 1		Disturbance 2	
$\approx 15°$	$\approx 30°$	at time step 7	at time step 14
10/10	10/10	10/10	10/10

TABLE I: Success rate for recovery from two types of disturbances.

brings the half-cylinder from 0 degree (Fig 3.A) to 90 degrees (Fig 3.B). Then a manually-designed controller flips the half-cylinder completely (Fig 3.DE). The closed-loop controller stops early once the goal region has been reached (Fig 3.C), followed by the same manually-designed controller. The open-loop controller is already very robust during execution, so we tested some larger disturbances to show the robustness of the closed-loop controller. We experimented two types of disturbances. One is to force the half-cylinder to rotate clockwise using a human hand (Fig 3.F). We can easily fail the open-loop controller by holding the half-cylinder long enough so that the controller finishes open-loop execution. We found that the closed-loop controller always recovers when the disturbances are no more than 30 degrees. The success rate is 100% (20 out of 20). We observe that the contact modes do not change under small disturbances (Fig 4 Right). We also observe that the controller can sometimes recover from very large disturbances that violate the assumptions we made when designing the trajectory, e.g., the disturbance is more than 30 degrees and the right finger is on the flat surface of the half-cylinder. The other type of disturbance is to "accidentally" open the gripper for 2 cm at a certain time step (before opening: Fig 3.G, after opening: Fig 3.H). This tests the local multi-contact stabilizing controller. The success rate is also 100% (20 out of 20). The contact modes change when the disturbance happens (Fig 4 Right). The controller can also recover from large disturbances that violate our design assumptions, e.g., the gripper opens 4 cm at a certain time step.

VIII. Discussion and Conclusion

We have described a locally robust feedback design algorithm for dexterous manipulation. We have shown on hardware that the algorithm can recover from large external disturbances.

There are some limitations. First, if the size of the half-cylinder or the object shape changes, we need to analyze the dynamics, rerun the trajectory optimization, and build the stabilizing controllers offline. Building a large library for many shapes and various sizes is a possible solution to robust manipulation. Second, we used AR-tag to track the pose of the object. The estimation for the contact points between the gripper and the object are very rough, which causes some problems sometimes. In practice, perception is important for manipulation. Third, the algorithm does not work for some types of manipulation, for example throwing or sorting a deck of cards. Enabling robots to do more complicated tasks using model-based approaches still remains to be explored.

ACKNOWLEDGMENT

This work was supported by Air Force/Lincoln Laboratory Award No. PO# 7000374874, and Lockheed Martin Corporation Award No. RPP2016-002. The first author would like to thank Twan Koolen for helpful discussions about dynamics and humanoid robots, Sadra Sadraddini for helpful discussions on polytopic trees, Tao Pang for setting up the Kuka robot, Greg Izatt for providing initial code for controlling the Kuka robot, and Pete Florence for helping with the AR-tag and the Kinect and for tele-operating the Kuka robot.

References

[1] M. T. Mason, "Toward robotic manipulation," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 1, pp. 1–28, 2018.

[2] Z. Li and S. S. Sastry, "Task-oriented optimal grasping by multifingered robot hands," *IEEE Journal on Robotics and Automation*, vol. 4, no. 1, pp. 32–44, 1988.

[3] A. T. Miller, S. Knoop, H. I. Christensen, and P. K. Allen, "Automatic grasp planning using shape primitives," in *Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on*, vol. 2. IEEE, 2003, pp. 1824–1829.

[4] N. Chavan-Dafle, R. Holladay, and A. Rodriguez, "In-hand manipulation via motion cones," in *Robotics: Science and Systems*, 2018.

[5] J. Zhou, R. Paolini, J. A. Bagnell, and M. T. Mason, "A convex polynomial force-motion model for planar sliding: Identification and application," in *Robotics and Automation (ICRA), 2016 IEEE International Conference on*. IEEE, 2016, pp. 372–377.

[6] N. C. Dafle, A. Rodriguez, R. Paolini, B. Tang, S. S. Srinivasa, M. Erdmann, M. T. Mason, I. Lundberg, H. Staab, and T. Fuhlbrigge, "Extrinsic dexterity: In-hand manipulation with external forces," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 1578–1585.

[7] S. Levine, N. Wagener, and P. Abbeel, "Learning contact-rich manipulation skills with guided policy search," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*. IEEE, 2015, pp. 156–163.

[8] S. Levine, P. Pastor, A. Krizhevsky, J. Ibarz, and D. Quillen, "Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection," *The International Journal of Robotics Research*, vol. 37, no. 4-5, pp. 421–436, 2018.

[9] C. Finn and S. Levine, "Deep visual foresight for planning robot motion," in *Robotics and Automation (ICRA), 2017 IEEE International Conference on*. IEEE, 2017, pp. 2786–2793.

[10] V. Kumar, E. Todorov, and S. Levine, "Optimal control with learned local models: Application to dexterous manipulation," in *Robotics and Automation (ICRA), 2016 IEEE International Conference on*. IEEE, 2016, pp. 378–383.

[11] A. Zeng, S. Song, J. Lee, A. Rodriguez, and T. Funkhouser, "Tossingbot: Learning to throw arbitrary objects with residual physics," *arXiv preprint arXiv:1903.11239*, 2019.

[12] M. Andrychowicz, B. Baker, M. Chociej, R. Jozefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray *et al.*, "Learning dexterous in-hand manipulation," *arXiv preprint arXiv:1808.00177*, 2018.

[13] S. Kuindersma, R. Deits, M. Fallon, A. Valenzuela, H. Dai, F. Permenter, T. Koolen, P. Marion, and R. Tedrake, "Optimization-based locomotion planning, estimation, and control design for the atlas humanoid robot," *Autonomous Robots*, vol. 40, no. 3, pp. 429–455, 2016.

[14] A. Majumdar and R. Tedrake, "Funnel libraries for real-time robust feedback motion planning," *The International Journal of Robotics Research*, vol. 36, no. 8, pp. 947–982, 2017.

[15] S. Sadraddini and R. Tedrake, "Sampling-based polytopic trees for approximate optimal control of piecewise affine systems," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 7690–7696.

[16] J. Shi, J. Z. Woodruff, P. B. Umbanhowar, and K. M. Lynch, "Dynamic in-hand sliding manipulation," *IEEE Transactions on Robotics*, vol. 33, no. 4, pp. 778–795, 2017.

[17] J.-C. Ryu, F. Ruggiero, and K. M. Lynch, "Control of nonprehensile rolling manipulation: Balancing a disk on a disk," *IEEE Transactions on Robotics*, vol. 29, no. 5, pp. 1152–1161, 2013.

[18] P. Umbanhowar, T. H. Vose, A. Mitani, S. Hirai, and K. M. Lynch, "The effect of anisotropic friction on vibratory velocity fields," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. IEEE, 2012, pp. 2584–2591.

[19] T. H. Vose, P. Umbanhowar, and K. M. Lynch, "Manipulation with vibratory velocity fields on a tilted plate," in *Automation Science and Engineering (CASE), 2012 IEEE International Conference on*. IEEE, 2012, pp. 942–949.

[20] J. Z. Woodruff and K. M. Lynch, "Planning and control for dynamic, nonprehensile, and hybrid manipulation tasks," in *Robotics and Automation (ICRA), 2017 IEEE International Conference on*. IEEE, 2017, pp. 4066–4073.

[21] F. R. Hogan, E. R. Grau, and A. Rodriguez, "Reactive planar manipulation with convex hybrid mpc," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 247–253.

[22] S. M. LaValle, *Planning algorithms*. Cambridge university press, 2006.

[23] M. Likhachev, D. I. Ferguson, G. J. Gordon, A. Stentz, and S. Thrun, "Anytime dynamic a*: An anytime, replanning algorithm." in *ICAPS*, 2005, pp. 262–271.

[24] A. C. Shkolnik, "Sample-based motion planning in high-dimensional and differentially-constrained systems," MASSACHUSETTS INST OF TECH CAMBRIDGE COMPUTER SCIENCE AND ARTIFICIAL , Tech. Rep., 2010.

[25] M. Posa, C. Cantu, and R. Tedrake, "A direct method for trajectory optimization of rigid bodies through contact," *The International Journal of Robotics Research*, vol. 33, no. 1, pp. 69–81, 2014.

[26] H. Dai, A. Valenzuela, and R. Tedrake, "Whole-body motion planning with centroidal dynamics and full kinematics," in *Humanoid Robots (Humanoids), 2014 14th IEEE-RAS International Conference on*. IEEE, 2014, pp. 295–302.

[27] I. Mordatch, E. Todorov, and Z. Popović, "Discovery of complex behaviors through contact-invariant optimization," *ACM Transactions on Graphics (TOG)*, vol. 31, no. 4, p. 43, 2012.

[28] M. Toussaint, "A novel augmented lagrangian approach for inequalities and convergent any-time non-central updates," *arXiv preprint arXiv:1412.4329*, 2014.

[29] A. K. Valenzuela, "Mixed-integer convex optimization for planning aggressive motions of legged robots over rough terrain," Ph.D. dissertation, Massachusetts Institute of Technology, 2016.

[30] Y. Tassa, N. Mansard, and E. Todorov, "Control-limited differential dynamic programming," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*. IEEE, 2014, pp. 1168–1175.

[31] Y. Tassa, T. Erez, and E. Todorov, "Synthesis and stabilization of complex behaviors through online trajectory optimization," in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*. IEEE, 2012, pp. 4906–4913.

[32] D. Dolgov, S. Thrun, M. Montemerlo, and J. Diebel, "Path planning for autonomous vehicles in unknown semi-structured environments," *The International Journal of Robotics Research*, vol. 29, no. 5, pp. 485–501, 2010.

[33] X. Zhang, A. Liniger, A. Sakai, and F. Borrelli, "Autonomous parking using optimization-based collision avoidance," in *2018 IEEE Conference on Decision and Control (CDC)*. IEEE, 2018, pp. 4327–4332.

[34] B. D. Anderson and J. B. Moore, *Optimal control: linear quadratic methods*. Courier Corporation, 2007.

[35] M. Baoti, F. J. Christophersen, and M. Morari, "Constrained optimal control of hybrid systems with a linear performance index," *IEEE Transactions on Automatic Control*, vol. 51, no. 12, pp. 1903–1919, 2006.

[36] F. J. Christophersen, M. Baotić, and M. Morari, "Optimal control of piecewise affine systems: A dynamic programming approach," in *Control and Observer Design for Nonlinear Finite and Infinite Dimensional Systems*. Springer, 2005, pp. 183–198.

[37] F. Borrelli, M. Baotić, A. Bemporad, and M. Morari, "Dynamic programming for constrained optimal control of discrete-time linear hybrid systems," *Automatica*, vol. 41, no. 10, pp. 1709–1721, 2005.

[38] M. Barić, P. Grieder, M. Baotić, and M. Morari, "An efficient algorithm for optimal control of pwa systems with polyhedral performance indices," *Automatica*, vol. 44, no. 1, pp. 296–301, 2008.

[39] A. Bemporad, F. Borrelli, and M. Morari, "Optimal controllers for hybrid systems: Stability and piecewise linear explicit form," in *Proceedings of the 39th IEEE Conference on Decision and Control (Cat. No. 00CH37187)*, vol. 2. IEEE, 2000, pp. 1810–1815.

[40] ——, "Piecewise linear optimal controllers for hybrid systems," in *Proceedings of the 2000 American Control Conference. ACC (IEEE Cat. No. 00CH36334)*, vol. 2. IEEE, 2000, pp. 1190–1194.

[41] ——, "On the optimal control law for linear discrete time hybrid systems," in *International workshop on hybrid systems: computation and control*. Springer, 2002, pp. 105–119.

[42] D. Mayne and S. Rakovic, "Optimal control of constrained piecewise affine discrete time systems using reverse transformation," in *Proceedings of the 41st IEEE Conference on Decision and Control, 2002.*, vol. 2. IEEE, 2002, pp. 1546–1551.

[43] L. Rodrigues, "Dynamic output feedback controller synthesis for piecewise-affine systems," Ph.D. dissertation, Stanford University, 2002.

[44] M. Lazar, "Model predictive control of hybrid systems: Stability and robustness," Ph.D. dissertation, Eindhoven: Technische Universiteit Eindhoven, 2006.

[45] W. Han and R. Tedrake, "Feedback design for multi-contact push recovery via lmi approximation of the piecewise-affine quadratic regulator," in *Humanoid Robotics (Humanoids), 2017 IEEE-RAS 17th International Conference on*. IEEE, 2017, pp. 842–849.

[46] P. Zhao, S. Mohan, and R. Vasudevan, "Optimal control for nonlinear hybrid systems via convex relaxations," *arXiv preprint arXiv:1702.04310*, 2017.

[47] W. Han and R. Tedrake, "Controller synthesis for discrete-time hybrid polynomial systems via occupation measures," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 7675–7682.

[48] T. Marcucci, R. Deits, M. Gabiccini, A. Bicchi, and R. Tedrake, "Approximate hybrid model predictive control for multi-contact push recovery in complex environments," *Under Review*, 2017.

[49] D. Stewart and J. C. Trinkle, "An implicit time-stepping scheme for rigid body dynamics with coulomb friction," in *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, vol. 1. IEEE, 2000, pp. 162–169.

[50] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical programming*, vol. 106, no. 1, pp. 25–57, 2006.

[51] S. Kajita, H. Hirukawa, K. Harada, and K. Yokoi, *Introduction to humanoid robotics*. Springer, 2014, vol. 101.

[52] S. Kajita, F. Kanehiro, K. Kaneko, K. Fujiwara, K. Harada, K. Yokoi, and H. Hirukawa, "Biped walking pattern generation by using preview control of zero-moment point," in *2003 IEEE International Conference on Robotics and Automation (Cat. No. 03CH37422)*, vol. 2. IEEE, 2003, pp. 1620–1626.

[53] R. Tedrake, I. R. Manchester, M. Tobenkin, and J. W. Roberts, "Lqr-trees: Feedback motion planning via sums-of-squares verification," *The International Journal of Robotics Research*, vol. 29, no. 8, pp. 1038–1052, 2010.

[54] M. Fiala, "Artag, a fiducial marker system using digital techniques," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 2. IEEE, 2005, pp. 590–596.

[55] P. E. Gill, W. Murray, and M. A. Saunders, "Snopt: An sqp algorithm for large-scale constrained optimization," *SIAM review*, vol. 47, no. 1, pp. 99–131, 2005.