Relaxed-Rigidity Constraints: In-Grasp Manipulation using Purely Kinematic Trajectory Optimization

Balakumar Sundaralingam and Tucker Hermans
University of Utah Robotics Center and the School of Computing
University of Utah, Salt Lake City, UT, USA
Email: \{bala, thermans\}@cs.utah.edu

Abstract—This paper proposes a novel approach to performing in-grasp manipulation planning: the problem of moving an object with reference to the palm from an initial pose to a goal pose without breaking or making contacts. Our method to perform in-grasp manipulation uses kinematic trajectory optimization which requires no knowledge of dynamic properties of the object or the robot. We define a cost function that attempts to maintain the initial grasp points, while relaxing the constraint that the contacts between finger and object remain rigid. Hence, we name this new formulation relaxed-rigidity constraints. We implement our approach on an Allegro robot hand and perform experiments on 10 objects from the YCB dataset. However, the implementation would work for any object the robot can grasp. We perform thorough analysis and compare to alternative optimization formulations. Our method reaches the desired object pose with a median position error of 13mm across all of the 500 trials without ever dropping the object.

I. INTRODUCTION AND MOTIVATION

The problem of robotic in-hand manipulation—changing the relative pose between a robot hand and object, without placing the object down—remains largely unsolved. Research in in-hand manipulation has focused largely on using full knowledge of the mechanical properties of the objects of interest in finding solutions [1] [12] [22] [23]. This reliance on object specific modeling makes in-hand manipulation expensive and sometimes infeasible in real-world scenarios, where robots may lack high-fidelity object models. Learning-based approaches to the problem have also been proposed [20] [31]; however, these methods require significant experience with the object of interest to work and learn only a single motion primitive (e.g. movement to a specific goal pose). Solving the general in-hand manipulation problem using real world robotic hands will require a variety of manipulation skills [3]. As such, we focus on a subproblem of in-hand manipulation: in-grasp manipulation where the robot moves an object under grasp to a desired pose without changing the initial grasp. We explore a purely kinematic planning approach for in-grasp manipulation motivated by recent successes in kinematic grasp planning [6] [7].

Having a successful in-grasp manipulation planner would allow robots to change a grasped object’s pose without requiring full arm movement or complex finger gaiting [17]. Many tasks requiring a change in relative pose between the hand and the grasped object do not require a large workspace and can be performed without changing the grasp. Example tasks include turning a dial, reorienting objects for insertion, or assembling small parts such as watch gears. This is especially beneficial in cluttered environments where small movements would be preferred. This paper addresses the in-grasp task with a purely kinematic planning approach. By attempting to maintain the contact points from the initial grasp during manipulation, we do not require any detailed models of the grasped object.

The in-grasp manipulation problem is under-actuated, as the object’s states are not fully or directly controllable. As such, it does not immediately offer a kinematic solution. A naive approach would be to model all contacts between the object and robot as rigid links and plan for a desired object pose as if the robot were a parallel mechanism. However, in most robotic hands, the fingers have fewer degrees of freedom (DOF) than necessary to control a 6 DOF world pose. Thus, we introduce a novel cost function which relaxes the rigidity constraints between the object and fingers. The cost function penalizes the robot fingertips for changing the relative positions and orientations between each other from those used in the initial grasp. We name this cost function the relaxed-rigidity constraint. We combine relaxed-rigidity constraints for all fingers with cost terms that encourage the object's movement to the desired pose. This combined cost function defines the objective for our
purely kinematic trajectory optimization. The result allows for small position and orientation changes at the contact locations, while maintaining a stable grasp as the object moves toward the desired pose. Fig. [1] shows an example trajectory from our planner.

Our approach to in-grasp manipulation directly solves for a joint space trajectory to reach a task space goal, in contrast to previous methods [21 23] which rely on separate inverse kinematic (IK) solvers to obtain joint space trajectories. Our direct solution is attractive, as IK solutions become complex when a robot is under-actuated in terms of the dimensions of the task space (i.e. the end-effector of a 4 joint manipulator cannot reach all orientations in a 6 dimensional task space for a given position). Our approach additionally handles hard constraints on the robot’s joint positions and velocities. The problem is efficiently solved as a direct optimization using a sequential quadratic programming (SQP) solver. Our method allows for changing the object’s pose without the need to know the dynamic properties of the object or the contact forces on the fingers. Currently, we perform the planned trajectory on the robot with no feedback of the object’s pose.

This paper makes the following contributions validated with real-world experiments:

1) We demonstrate that a purely kinematic trajectory optimization sufficiently solves a large set of in-grasp manipulation tasks with a real robot hand.

2) We enable this kinematic solution by introducing a novel relaxation of rigid-contact constraints to a soft constraint on rigidity expressed as a cost function. We name this the relaxed-rigidity constraint.

3) Our method directly solves for joint configurations at all time steps, without the need of a separate inverse kinematics solver, a novel contribution over previous trajectory optimization approaches for in-hand manipulation (e.g. [23]).

4) We are the first to extensively validate an in-grasp manipulation planner on a real robot hand. We do so with multiple objects from the YCB dataset [5] and introduce relevant error metrics, paving the way for a unified testing scheme in future works (c.f. Section IV).

We organize the remainder of the paper as follows. In-hand manipulation research related to our approach is discussed in Section II. We follow this with a formal definition of the problem and a detailed explanation of our approach in Section III. We then discuss implementation details and define our experimental protocol in Section IV. We analyze the results of extensive robot experiments in Section V followed by a final discussion of conclusions and future work in Section VI.

II. RELATED WORK

In-hand manipulation has been studied extensively [4 8 15 22]. The topic is often referred to as dexterous manipulation (e.g. [12]) or fine manipulation (e.g. [17]). We choose the term in-hand manipulation to highlight the fact that the operations happen with respect to the hand and not the world or other parts of the robot. We believe that dexterity can be leveraged for a number of tasks, which do not fundamentally deal with in-hand manipulation, and that a robot can finely manipulate objects without the need for multi-fingered hands or grasping. This section covers those methods that are most relevant to our approach and does not discuss in detail methods for finger gaiting (e.g. [17 28]) or dynamic in-hand manipulation [2 30].

Li and colleagues [22] developed a computed torque controller for coordinated movement of multiple fingers on a robot hand. The controller takes as input a desired object motion and contact forces and outputs the set of finger torques necessary to create this change. The controller requires models of the object dynamics (mass and inertia matrix) in order to compute the necessary control commands. The authors demonstrate in simulation the ability for the controller to have a planar object follow a desired trajectory, when grasped between two fingers. Härtl [15] factors forces on objects and force to joint torque conversions to perform in-hand manipulation accounting for slippage and rolling. An analytical treatment of dynamic object manipulation is explored with ways for reducing the computations required.

Han et al. [12 13] attempt in-hand manipulation with rolling contacts and finger gaiting requiring knowledge of the object surface. They solve for Cartesian space finger-tip and object poses. Results for rolling contacts are demonstrated using flat fingertips to manipulate a spherical ball. The robot tracks the end-effector velocities using the manipulator Jacobian to determine the joint velocities.

Bicchi and Sorrentino [4] analyze the kinematics of rolling an object grasped between fingers. The authors present a planner for rolling a sphere between two large plates acting as fingers. This is achieved through creating a state feedback law of vector flow fields. All these early methods require extensive details about the object which is hard to obtain in the real world and is inefficient when attempting to manipulate novel objects.

In-hand manipulation research has diverged in terms of approaches. Mordatch et al. [23] formalize in-hand manipulation as an optimization problem. They solve for a task space trajectory and obtain joint space trajectories for the robot using an IK solver independent of their optimization. The trajectory optimization approach factors in force closure, but uses a joint level position controller to perform the manipulation assuming they have a perfect robot dynamics model to convert end-effector forces to positions. Experimental evaluation is shown only in simulation.

Similar to our approach, Hertkorn et al. [16] seek to find a trajectory to a desired object pose without changing the grasp. Their approach additionally solves for an initial grasp configuration to perform the desired motion in space. They discretize the problem by creating configuration space graphs for different costs and use a union of these graphs to choose a stable grasp. They perform an exhaustive search through this union of graphs to find the desired trajectory. Their approach does not scale, even to simple 3D problems, with multi-fingered robot hands as stated by the authors and they show no real-world results. Their method is computationally inefficient.
as the reported time for finding a single feasible trajectory was 60 minutes for a two fingered robot with 2 joints each. In contrast, we efficiently and directly solve for joint positions in the continuous domain using trajectory optimization.

Andrews and Kry [1] take a hierarchical approach to in-hand manipulation by splitting the problem into three phases: approach, actuation, and release. Their method uses an evolutionary algorithm to optimize the individual motions. This requires many forward simulations of the full dynamical system and does not leverage gradient information in the optimization. Another drawback, as stated by the authors, is that their approach cannot be applied to objects with complex geometry.

Hang et al. [14] explore grasp adaptation to maintain a stable grasp and compensate for slippage or external disturbances with tactile feedback. They use the object’s surface geometry to choose contact points for grasping and when performing finger gaiting to maintain a stable grasp. Their method could be used to obtain an initial stable grasp which could then be used in our approach to move the object to a desired pose.

Li et al. [21] use two KUKA arms to emulate in-hand manipulation with tactile and visual feedback to move objects to a desired pose. The use of flat contact surfaces limits the possible trajectories of the object. The use of a 7 joint manipulator as a finger also allows for reaching a larger workspace than common robotic hands which are mostly limited to 4 joint fingers. The evaluation is limited to position experiments and single axis orientation changes.

Rojas and Dollar [27] present a method to analyze the cinematic-motion of a hand with respect to a grasped object. This tool could be used to find feasible goal poses for an object without changing the current grasp similar to [16]. However the authors are motivated by designing dexterous robot hands and do not perform any planning with their technique.

Kumar et al. [19] examine the use of model-predictive control for a number of tasks including in-hand manipulation. They rely on hand synergies and full models of the robot and object dynamics to compute their optimal controllers. However, they recently built on this approach [20] and used machine learning to construct dynamics models for the object-hand system. These models could then be used to create a feedback controller to track a specific learned trajectory. They show results on a real robot hand with a high number of states and actuators. However, their method requires retraining to be used in manipulating a new object or moving to a new goal pose. Van Hoof et al. [31] use reinforcement learning to learn a policy for rolling an object in an under-actuated hand. The resulting policy leverages tactile feedback to adapt to different objects, however they must learn a new policy if they were to change the desired goal. Finally, while these learning-based methods show promise in rapidly converging to a desired controller, they still require multiple runs on the robot.

In contrast, we perform in-grasp manipulation on a real robot hand with novel objects, without requiring extensive object information or performing any iterative learning.

III. PROBLEM DEFINITION AND PROPOSED APPROACH

We define the problem of in-grasp manipulation as finding a trajectory of joint angles $\Theta = [\Theta_1, \ldots, \Theta_T]$ that moves the object from its initial pose $X_0$ at time 0 to a desired object pose $X_g$ at time $T$ without changing the grasp on the object. We address this problem under the following simple assumptions:

1) The object’s pose can only be affected by the robot and gravity, i.e. there are no external systems acting on the object.
2) The object is rigid.
3) The initial grasp is a stable grasp of the object.
4) The desired object pose is in the reachable workspace of the fingertips.

We formulate our solution as a nonlinear, non-convex constrained kinematic trajectory optimization problem:

$$\min_{\Theta} E_{obj}(\Theta_T, X_g) + k_1 \sum_{t=0}^{T-1} E_{obj}(\Theta_t, W_t) + k_2 \sum_{t=0}^{T} E_{pos}(\Theta_t) + k_3 \sum_{t=0}^{T} E_{or}(\Theta_t)$$

subject to:

$$\Theta_{min} \leq \Theta_t \leq \Theta_{max}, \forall t \in [0, T]$$

$$-\frac{\Theta_{t-1} - \Theta_t}{\Delta t} \leq \Theta_{max}, \forall t \in [1, T]$$

The first constraint enforces the joint limits of the robot hand, while the second inequality constraint limits the velocity of the joints to prevent rapid movements. The scalar weights, $k_1, k_2, k_3$, on each cost term allow us to tune the trade-off between the four cost components. $W_t$ defines the waypoint of the object pose at time $t$ computed automatically as described below.

In order to achieve a purely kinematic formulation, we plan with a number of approximations, which we validate with experiments in Sec V. We now describe the components of the cost function in detail.

A. Object Pose Cost

The first term in the cost function (Eq[1]), $E_{obj}(\Theta_T, X_g)$, is designed to minimize the euclidean distance between the planned object pose at the final time step $X_T$ to the desired final object pose $X_g$. Our kinematic trajectory optimization approach assumes no knowledge of the dynamic properties of the object, as such we can not directly simulate the object pose $X_t$ during our optimization. Instead, we leverage the fact that in-grasp manipulation assumes no breaking or making of contacts during execution, meaning, in the ideal case, contact points between the robot and object remain fixed.

In our approach we thus plan as if the contact point between the thumb-tip and the object is rigid. This allows us to define a reference frame for the object $X$ with respect to the thumb-tip $i$ such that the transformation between the thumb-tip and
the thumb at the desired object pose. The initial pose of the object with the grasp is shown in (a), the object has to reach the goal pose (b). Our approach models the thumb frame as rigidly attached to the object during the trajectory, while finger $f$ has a relaxed-rigidity constraint. The effect can be seen in (b), where the relative orientation and position between frames $i$ and $f$ have changed from the initial grasp at $t=0$. The $\dot{T}_iP_f$, $\ddot{T}_iP_f$, $c_0$ and $c_T$ terms from Sec. [III-B] are shown as green, blue, orange and red vectors respectively. $\theta_{i1:i3}$ are the joint angles of thumb and finger $f$.

We can now transform the desired object pose $X_g$ into a desired thumb pose $G_i$ in the world frame $w$. The cost function $E_{obj}$ can now be formally defined as

$$E_{obj}(\Theta_T, X_g) = ||w^T G_i - w^T F_i||_2^2$$

where, $w^T G_i = X_g \cdot X T_i$, and $w^T F_i = FK(\Theta_T, i)$ gives the pose of the thumb-tip $i$ with reference to the world frame. Thus by using the forward kinematics (FK) internally within the cost function we can directly solve for the joint angles of the thumb at the desired object pose.

The second term $\sum_{i=1}^{T-1} E_{obj} (\Theta_t, W_t)$ present in the cost function (Eq. [1]) encourages shorter paths to the desired pose. We define the waypoints $W_t$ for every time-step $t$ be linearly interpolated from the initial object pose to the desired object pose $X_g$, equally spaced across all timesteps. We weigh this term at a very low scale relative to the other cost terms, to encourage a shorter path as a linear path is not always guaranteed.

**B. Relaxed-Rigidity Constraints**

Since most robotic fingers are under-actuated with respect to possible 6 DOF fingertip poses, we can’t apply the same rigid-contact constraint with respect to the object pose, as we did for the thumb for all the remaining fingers. Doing so would reduce the reachable space for the remaining fingers, resulting in a smaller manipulable workspace for the object (manipulable workspace being the workspace covering all possible object poses for a given grasp). Instead, we relax the rigid-contact constraint for all other fingers in the grasp, allowing for a larger manipulable workspace. The remaining two terms in the cost function (Eq. [1]), $E_{pos}$ and $E_{or}$, define our novel relaxed-rigidity constraint. The combined effect of the terms encourages the fingertips to remain at the same contact points on the object throughout the trajectory.

We define the cost term $E_{pos}(\Theta_t)$ to maintain the initial relative positions between the thumb, $i$, and the remaining fingertips, $f$ throughout execution:

$$E_{pos}(\Theta_t) = \sum_{f=1}^{n} ||i^TP_f - i^TP_f||_2^2$$

where $i^TP_f = i^T \cdot FP(\Theta_t, f)$ defines the fingertip position for finger $f$ in the thumb frame $i$ at time $t$ and $FK_P(\Theta_t, f)$ computes the position of fingertip $f$ for joint configuration $\Theta_t$. Combined with the object pose cost, which moves the thumb towards the goal pose, this cost minimizes deviation from the initial grasp, while moving towards the goal pose.

The last cost term $E_{or}(\Theta_t)$ encourages the other fingers to maintain their relative orientation to the thumb to be the same as that in the initial grasp. Maintaining this cost across all three orientation dimensions, would again over-constrain the problem to the full rigidity constraint. We relax this constraint by introducing a weight vector $\psi$ which defines a relative preference for deviation in different orientation dimensions

$$E_{or}(\Theta_t) = \sum_{f=1}^{n} ||(c_f^T - c_f^T) \cdot \psi||_2^2$$

where $c_f^T = FK_{RPV}(\Theta, f)$ computes the roll, pitch, yaw of the unit vector between the thumb, $i$ and finger $f$ at time $t$. Fig. [2] illustrates the vectors used in the relaxed-rigidity constraints.

**IV. IMPLEMENTATION AND EXPERIMENTAL PROTOCOL**

We now describe important details relating to our implementation and the setup of our experiments.
A. Trajectory Generation and Robot Execution

Direct methods for trajectory optimization, such as sequential quadratic programming (SQP), have shown promising results in robotics [24, 25, 29]. We solve our trajectory optimization problem using SNOPT [11] an SQP solver designed for sparsely constrained problems. We run the solver with a maximum limit of 5000 iterations using analytical gradients for the costs and constraints. The computer used to run the solver and experiments is an Intel i7-4790K with 32 GB of RAM running Ubuntu 14.04 with ROS Indigo [26]. The robot used is the Allegro Hand, which has four fingers with 4 joints each. We solve for $T = 10$ time steps with each time step being $\Delta t = 0.167\text{s}$ long. We expand the obtained solution to a higher resolution of 100 time steps by linearly interpolating the joint trajectories. We limit the joint velocities to be less than $0.6\text{rad/s}$. Our approach has four weights-three on scaling the importance of each cost term-$k_1, k_2, k_3$ and a projection weight $\psi$ for the orientation cost term $E_{or}$.

The orientation cost $E_{or}$ reduces orientation changes along the weight vector $\psi$. Ideally, we would want to reduce the impact of any contact model on the manipulation task by having weights across all three dimensions. However, this would reduce the reachable workspace of the manipulation task as the Allegro hand is under-actuated with respect to the 6 DOF poses. Hence, we chose to reduce orientation changes along a single axis, which covers the largest workspace of fingertip positions. This is the $y$-axis with respect to the palm for the index, middle and ring fingers, making $\psi = [0 \ 1 \ 0]$ as seen in Fig. 5. We consider this weight as a trade off between allowing a larger manipulation workspace and enforcing smaller changes at contact points. We see in Sec. V-B that restricting orientation changes along one axis improves the position error over assuming a point contact model.

The remaining three weights model the relative importance between the cost terms of our optimization. We want the robot to always maintain contacts close to the initial grasp during manipulation, this is taken care of by large value for $k_2$. Keeping the initial orientation is less important, allowing $k_3$ to be less than $k_2$. The weight for waypoints $k_1$ should help guide the fingers to the goal pose, while being low enough to allow for non-linear trajectories when linear trajectories are not feasible. We examined various weights under this scaling and found $k_1 = 0.09, k_2 = 100, k_3 = 1$ to work well across a variety of trajectories and objects. For $k_2 < 1.0$, the hand dropped the object when unreachable object poses were given. The chosen weights, however, were able to maintain the object in-grasp while still moving the object towards the desired pose.

The allegro hand runs an on-board PD based joint position controller. Our open loop implementation consists of sending the joint space trajectory over 100 time steps at 60 Hz. Solving for a single trajectory takes approximately 13 seconds on average.

B. Comparison Methods

Relatively little empirical evaluation has been performed for in-hand manipulation on real robot hands. The lack of a common benchmarking scheme prohibits us from comparing directly with methods described in Sec. I. Hence, we compare our method with two formulations representing two extremes of contact models. The two methods are formulated as a trajectory optimization problem similar to our method with different cost functions. The first method assumes a rigid contact model between the object and the fingertips, with the cost function:

$$
\sum_{i=0}^{n} ||(w G_i - w_T F_i)||^2 + w_1 \sum_{t=0}^{T-1} \sum_{i=1}^{n} ||(w_T T_0 X_t T_1 - w F_i)||^2
$$

we name this method “IK-rigid”.

The second method assumes a point contact with friction model (“PC”) for the fingertips. That is we attempt to keep

$$
\sum_{i=0}^{n} ||(w G_i - w_T F_i)||^2 + w_1 \sum_{t=0}^{T-1} \sum_{i=1}^{n} \sum_{j=1}^{n} ||(w_T T_0 X_t T_1 - w F_i)||^2
$$

we name this method “IK-rigid”.

Fig. 3: Objects from the YCB dataset with labels below each of them used in our experiments.

Fig. 4: Example grasps tested with various objects in our method.

Fig. 5: Desired object centroid poses for all experiments. The colors-red, blue and green in each frame represent the x, y and z axes. The palm frame is represented by ‘O’ with arrows.
the fingertips fixed with respect to the object, while allowing the relative orientation to change. This is a simplification of the model formulated in [22] to the cost function:

$$\sum_{i=0}^{n} \left\| \left( w^T G_{t} - w^T F_{i} \right) \alpha \right\|^2_2 + w_1 \sum_{t=0}^{T-1} \sum_{i=0}^{n} \left\| \left( w^T G_{t} - w^T F_{i} \right) \cdot \alpha \right\|^2_2$$  (9)

where $w^T G_{t}$ defines the desired fingertip pose at time $t$. We generate the desired fingertip poses by computing the fingertip’s pose in the object frame from object pose way-points $W_t$, computed via linear interpolation in the same manner as our approach. Since point contact with friction assumes that orientation changes at the contact points do not affect the object pose, the cost function only applies to fingertip positions and hence $\alpha = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 \end{bmatrix}$. This method does not model the object explicitly in the optimization. As such, the object pose cannot be directly obtained from a joint space trajectory without execution. The weight is chosen as $w_1 = 10^{-2}$.

C. Experimental Protocol and Evaluation

We selected objects of different size, texture and shape from the YCB dataset [5], shown in Fig. [5] as a benchmarking set. The ten objects used are: screwdriver, Lego, fork, banana, spatula, toy plane, jello, tuna, apple, and orange. A variety of three-fingered grasps were performed across the objects to show the reachability of the proposed method; examples can be seen in Fig. [4].

The set of experiments consist of moving the object under grasp to a goal pose, with ten unique reachable goal poses and two initial grasps per object. Finding feasible desired poses given an initial grasp is a complex problem [16, 27] and we do not formalize a method to obtain them. Instead, we focus on obtaining trajectories to a reachable pose and not on finding reachable poses. We obtain goal poses by having a human move the object in-grasp to the desired pose with the robot in gravity compensation mode. Any other method could be used to obtain desired poses. We show the distribution of the desired poses in Fig. [5]. The Euclidean distance to the desired positions from the initial object positions, range from 0.8cm to 8.33cm.

The trajectories are generated from our method (“Relaxed-Rigidity”). The frame ‘O’ represents the current object pose and ‘G’ frame is the desired object pose. Tuna being heavier than all other objects has a larger error due to the PD controller of the hand being insufficient to counteract the gravitational forces. Banana, having a complex surface also shows a larger error than objects with a flat surface. Markers used for ground truth collection only. Additional execution of different objects are shown at https://youtu.be/Gn-yMRjbmPE.
with a mean of 4.87cm. Desired poses with small positional change have a large orientation change. One trajectory for each goal pose was generated. To account for variation in execution and evaluate robustness, 5 trials are run for each trajectory giving a total of 50 trials per object. The difference in initial position between trials has a mean error of 0.59cm with an associated variance of 0.09cm. A total of 2000 trials are run across different methods to evaluate our proposed method.

The ground truth of the object pose is obtained using Aruco markers [9,10]. The initial pose of the object is obtained by placing the object in the hand and forming a grasp manually. Once the grasp is set, the joint angles are recorded and the object pose with respect to the palm link is obtained using the Aruco markers. We align the object with the initial pose used for trajectory generation using the markers and robot forward kinematics. Execution of all trials are recorded (video, robot frames, and object poses). All associated data is available (https://robot-learning.cs.utah.edu/project/in_hand_manipulation) to facilitate direct comparison.

We define the following error metrics for evaluating in-grasp manipulation. The position error is computed as the Euclidean distance between the reached position and the desired position of the object. Additionally, we report position error normalized with respect to the length of the trajectory, computed as the Euclidean distance between the initial and desired object positions. We report this normalized position error as “Position Error%”.

The second metric measures the final orientation error, calculated using quaternions, as the difference between rotation frames is not well defined using Euler angles [18].

$$\text{err}_{orient} = 100 \times \frac{\min(||q_d - q||_2, ||q_d + q||_2)}{\sqrt{2}}$$

(10)

where $q_d$ is the unit quaternion of the desired object pose and $q$ is the unit quaternion of the object pose reached. This error is in the range $[0, \sqrt{2}]$ and hence normalized with $\sqrt{2}$ and stated as “Orientation error%”. Finally, where appropriate, we report as failed attempts trials where the robot dropped the object during execution.

V. RESULTS

For every trajectory that is run on the robot, the position error and orientation error is recorded. The errors are plotted as a box plot (showing first quartile, median error, third quartile) with whiskers connecting the extreme values. The following subsections discuss the effectiveness of our method with respect to different criteria. In all plots results correspond to objects grasped with three fingers, unless otherwise stated.

A. Convergence of Optimization to Desired Object Pose

We first compare the results of different cost formulation within the trajectory optimization framework offline. Four methods were tested: (1) “IK-rigid” described in Sec. IV-B (2) “relaxed-position” which uses our proposed relaxed-rigidity position cost, but not the orientation cost or the waypoint cost; (3) “relaxed-position-orientation” which uses the full relaxed rigidity cost formulation with the relative position and orientation costs but not the waypoint cost; and (4) “Relaxed-Rigidity” which uses all our costs. We do not show offline results for the “PC” method as computing the object pose from the solution is not possible as the optimization does not internally simulate the object’s pose. However, we compare to this method in our real-robot experiments below.

The position error between the desired object pose and the final object pose obtained from the trajectory optimization are plotted in Fig. 7. It is evident that “IK-rigid” has difficulty reaching the desired object position, a result of the problem being over-constrained. The “relaxed-position” method incurs the least error, as there is no cost imposed on the orientation on the relaxed finger constraints. However, all relaxed-constraint approaches perform quite similarly. This analysis clearly shows how the “IK-rigid” method has a limited workspace, as such we do not report experimental results for this method on the actual robot.

B. Reaching Desired Object Poses

The position error and orientation error for all trials across all objects are shown in Fig. 8. Our method has the lowest median position error across all objects. The maximum error across all objects is also much smaller for our method than assuming a point contact model with friction. The “Position Error%” plot shows that our method—“Relaxed-Rigidity” is closer to the desired pose than the initial pose for all trials with a maximum error of 75%. In contrast “PC” obtains error greater than 100% for several trials, showing the object is moving further away from the desired pose than at the initial
We presented a generic in-grasp manipulation planner, which given only the initial joint angles, the joint limits, and the initial object pose, solves for a joint-level trajectory to move the object to a desired goal pose. The proposed method is implemented and experimentally validated on a real robot hand with ground truth error analysis. The results show that our relaxed-rigidity constraint allows for lower theoretical position error than a full rigidity constraint, and better real-world performance than assuming a point contact model.

Solving for the joint positions directly from the desired task space object pose, allowed us to implicitly solve the inverse kinematics problem. This enabled our single implementation to be successful for a diverse set of objects and grasps. A thorough analysis of the errors shows that high accuracy can be achieved using a kinematic only trajectory optimization approach to in-grasp manipulation, while never dropping the object.

Future work will involve adding object pose feedback to reduce the obtained errors. Though we did not encounter any collisions between the fingers during execution, we will explore adding collision checking to our optimization framework. Alternative smoothing costs to linear interpolation will also be explored.


