Autonomous Robots
https://doi.org/10.1007/s10514-018-9772-z

Relaxed-rigidity constraints: kinematic trajectory optimization and collision avoidance for in-grasp manipulation

Balakumar Sundaralingam[1] · **Tucker Hermans**[1]

Received: 1 December 2017 / Accepted: 22 May 2018

Abstract
This paper proposes a novel approach to performing in-grasp manipulation: 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. We implement our approach on an Allegro robot hand and perform thorough experiments on ten objects from the YCB dataset. The proposed method is general enough to generate motions for most objects the robot can grasp. Experimental results support the feasibillty of its application across a variety of object shapes. We explore the adaptability of our approach to additional task requirements by including collision avoidance and joint space smoothness costs. The grasped object avoids collisions with the environment by the use of a signed distance cost function. We reduce the effects of unmodeled object dynamics by requiring smooth joint trajectories. We additionally compensate for errors encountered during trajectory execution by formulating an object pose feedback controller.

Keywords Dexterous manipulation · Trajectory optimization · Motion planning

1 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 (Li et al. 1989; Mordatch et al. 2012; Han and Trinkle 1998; Andrews and Kry 2013). 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 (Kumar et al. 2016; Hoof et al. 2015); 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 (Bicchi 2000). 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 (Ciocarlie et al. 2007; Carpin et al. 2016).

Giving robots the ability to perform in-grasp manipulation would allow for changing a grasped object's pose without requiring full arm movement or complex finger gaiting (Hong et al. 1990). 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 to avoid collisions. In this paper, we propose a kinematic planner for in-grasp manipulation through trajectory optimization. The proposed planner gives a joint space trajectory that would move the object to the desired pose without losing grasp of the object. By attempting to maintain the contact points from

This is one of several papers published in *Autonomous Robots* comprising the "Special Issue on Robotics Science and Systems".

✉ Balakumar Sundaralingam
bala@cs.utah.edu

[1] University of Utah Robotics Center and the School of Computing, University of Utah, Salt Lake City, UT, USA

Published online: 05 June 2018

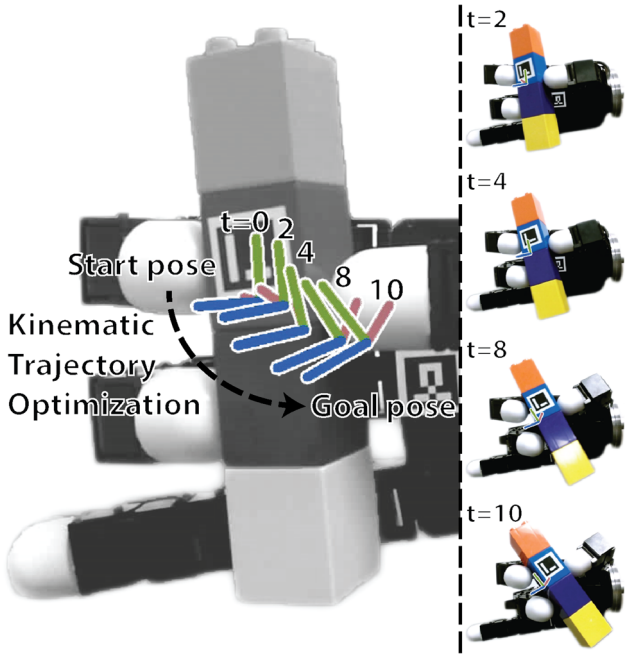


Fig. 1 Example trajectory produced by our method executed on the Allegro hand. The trajectory moves the Lego from its initially grasped pose to a desired pose. The robot follows the joint space trajectory produced from our trajectory optimizer with a PD based joint position controller. The images on the right show frames from execution where t refers to the timestep

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. This 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. Figure 1 shows an example trajectory from our planner. This kinematic planner successfully performed in-grasp manipulation with 10 objects across 500 trials without dropping the object.

Our approach to in-grasp manipulation directly solves for a joint space trajectory to reach a task space goal, in contrast to previous methods (Mordatch et al. 2012; Li et al. 2013) 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. Solving directly in the joint space also allows us to have costs in the input space such as smooth joint acceleration between time-steps to allow smooth operation of the robot during manipulation. The use of Trajectory optimization also allows for using advancements in collision-free manipulator motion planning (Schulman et al. 2014) to our in-grasp manipulation problem and we show how our planner can avoid collisions with the environment during manipulation. In addition, we compensate for error in trajectory execution online by incorporating an object pose feedback control scheme.

Our "Relaxed-Rigidity" planner 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. Mordatch et al. 2012).
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 (Calli et al. 2015) and introduce relevant error metrics, paving the way for a unified testing scheme in future works (c.f. Sect. 5).

This articles makes the following contributions over our previous work (Sundaralingam and Hermans 2017):

1. We introduce a joint acceleration cost to prefer smooth joint space paths, leading to lower object dynamics excitation.

2. We enable collision-free manipulation planning of the object in cluttered environments by including a signed distance cost function.
3. We compensate for error online during trajectory execution through an object-pose feedback controller.

We organize the remainder of the paper as follows. We discuss in-hand manipulation research related to our approach in Sect. 2. We follow this with a formal definition of the in-grasp manipulation problem and a detailed explanation of our in-grasp planner in Sect. 3. We present our extensions over the initial planner in Sect. 4. We then discuss implementation details and define our experimental protocol in Sect. 5. We analyze the results of extensive robot experiments in Sect. 6. We discuss the limitations of our approach in Sect. 7 and conclude in Sect. 8.

2 Related work

In-hand manipulation has been studied extensively (Li et al. 1989; Bicchi and Sorrentino 1995; Fearing 1986; Härtl 1995). The topic is often referred to as dexterous manipulation (e.g. Han and Trinkle 1998) or fine manipulation (e.g. Hong et al. 1990). 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. Hong et al. 1990; Rus 1992) or dynamic in-hand manipulation (Srinivasa et al. 2005; Bai and Liu 2014).

Salisbury and Craig (1982) explore grasping of objects with different hand designs. Salisbury and Roth (1983) explore gripping forces on grasped objects with three finger, three joint hand designs. Their work on force control with tendon driven articulated hands showed the need for dexterity near the end-effector for manipulation of grasped objects.

Li et al. (1989) 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 (1995) 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. (Han er al. 1997; Han and Trinkle 1998) 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 (1995) 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. (2012) 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. (2013) 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 min 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 (2013) 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. (2016) 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. (2013) 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.

Scarcia et al. (2015) perform in-grasp manipulation as a coordinated manipulation problem by adding arm motion planning. They assume a point contact with friction model between the object and the fingertips. They enumerate to obtain the reachability for an object pose. They do not perform extensive experiments with objects.

Rojas and Dollar (2016) present a method to analyze the kinematic-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 Hertkorn et al. (2013). However the authors are motivated by designing dexterous robot hands and do not perform any planning with their technique.

Kumar et al. (2014) 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 (Kumar et al. 2016) 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 if manipulating a new object or moving to a new goal pose. Hoof et al. (2015) 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 physical robot hand with novel objects, without requiring extensive object information or performing any iterative learning.

3 In-grasp manipulation planning through relaxed-rigidity constraints

We define the problem of in-grasp manipulation planning as finding a trajectory of joint angles $\boldsymbol{\Theta} \in [\Theta_1, \Theta_T]$ 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 fingertip contact points 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:

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

s.t.

$$\Theta_{min} \preceq \Theta_t \preceq \Theta_{max}, \forall t \in [0, T] \quad (2)$$

$$-\dot{\Theta}_{max} \preceq \frac{\Theta_{t-1} - \Theta_t}{\Delta t} \preceq \dot{\Theta}_{max}, \forall t \in [1, T] \quad (3)$$

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 Sect. 6. We now describe the components of the cost function in detail.

3.1 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

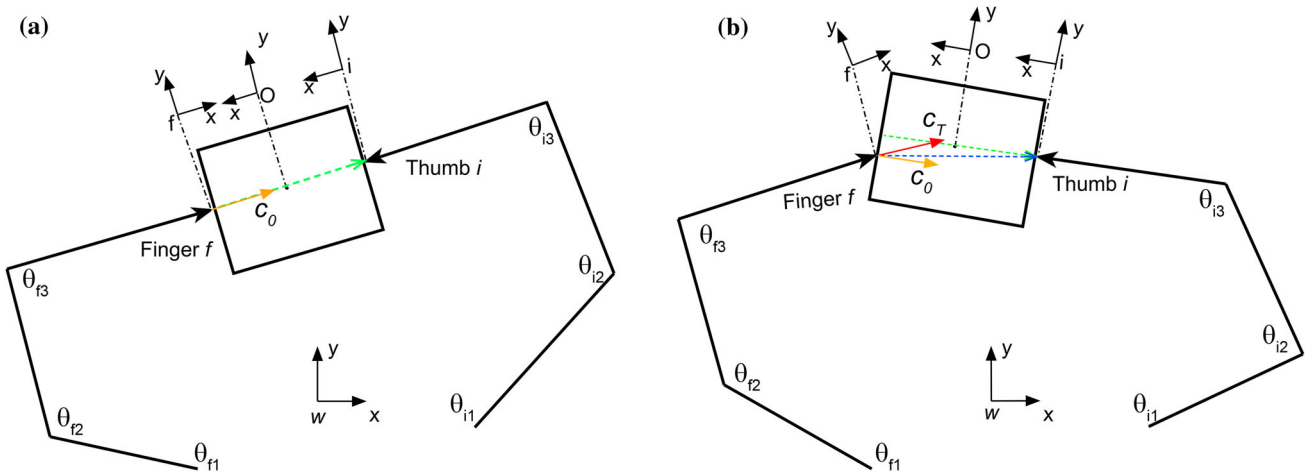


Fig. 2 Depiction of a trajectory optimization solution at the initial and final time steps. The thumb-tip frame is shown as frame i, finger f's tip frame is f, and w defines the world frame. The initial pose of the object with the grasp is shown in **a**, the object has to reach the goal pose (**b**). 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 ${}^{i}_{0}P_{f}$, ${}^{i}_{T}P_{f}$, c_0 and c_T terms from Sect. 3.2 are shown as green, blue, orange and red vectors respectively. $\theta_{\text{i1-i3}}$, $\theta_{\text{f1-f3}}$ are the joint angles of thumb and finger f. **a** (t = 0) **b** (t = T)

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[1] 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 object remains fixed during execution. As the thumb-tip moves with respect to the world frame w, we compute the transform to the object frame as

$$ {}^{w}T_{X} = {}^{w}T_{i} \cdot {}^{i}T_{X} \tag{4} $$

where the superscript refers to the reference frame and the subscript to the target frame. The object's transformation matrix is represented by ${}^{i}T_{X}$ with reference to thumb-tip i.

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

$$ E_{obj}(\Theta_T, X_g) = ||X_g \cdot {}^{X}T_{i} - FK(\Theta_T, i)||_2^2 \tag{5} $$

where, $X_g \cdot {}^{X}T_{i}$ and $FK(\Theta_T, i)$ 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 $\sum_{t=0}^{T-1} E_{obj}(\Theta_t, W_t)$ 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.

3.2 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.

[1] The choice of thumb is arbitrary and made only to clarify the discussion. Any fingertip could be chosen to define the reference frame for the object.

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:

$$E_{pos}(\Theta_t) = \sum_{f=1}^{n} || {}^{i}_{0}P_f - {}^{i}T_w \cdot FK_P(\Theta_t, f)||_2^2 \quad (6)$$

where ${}^{i}T_w \cdot FK_P(\Theta_t, f)$ 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 Θ_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 ψ which defines a relative preference for deviation in different orientation dimensions

$$E_{or}(\Theta_t) = \sum_{f=1}^{n} ||(FK^{i}_{RPY}(\Theta, f) - c_0^f) \cdot \psi||_2^2 \quad (7)$$

where $FK^{i}_{RPY}(\Theta, f)$ computes the roll, pitch, yaw of the unit vector between the thumb, i and finger f at time t. Figure 2 illustrates the vectors used in the relaxed-rigidity constraints.

4 Extensions

In this section we introduce two extensions to our relaxed-rigidity trajectory planner-joint acceleration smoothness and collision avoidance. We additionally propose an object pose feedback controller to compensate for errors encountered during execution of the in-grasp plan.

4.1 Joint acceleration

We find that the linear interpolation cost term

$$\sum_{t=0}^{T-1} E_{obj}(\Theta_t, W_t)$$

in Eq. 1 aids our trajectory optimization in finding a path to the desired object pose; however, it imposes two limitations:

1. The planner prefers a linear object path to the desired pose which may not always be possible.
2. The object velocity prefers to be constant during the manipulation which might cause sudden jerk of the object and thereby the joint control.

We explore an alternative cost which prefers smooth paths in the joint space. We minimize the acceleration between time steps following the sum of squares formulation from Toussaint (2017). This allows for smoother paths, while also not encouraging the object to follow a linear path to the desired pose. We replace the linear waypoint interpolation cost term with the following cost term,

$$\alpha_1 \sum_{t=0}^{T+1} (||\Theta_{t-2} - 2\Theta_{t-1} + \Theta_t||_2^2 \quad (8)$$

where we force $\Theta_{T+1} = \Theta_T$, and $\Theta_{-1} = \Theta_{-2} = \Theta_0$. We discuss the empirical effects of this in Sect. 6.2.

4.2 Collision avoidance

As proposed in Sect. 3 our planner does not avoid collisions between the object and the robot palm or the object and the environment. We now propose adding an obstacle-based cost function to the optimization in order to obtain collision-free plans while moving the object to the desired pose. We use signed distance functions to measure the distance between the grasped object and the environment motivated by other trajectory optimization approaches for motion planning (Zucker et al. 2013; Schulman et al. 2014).

The signed distance computes the shortest distance between a point p and the mesh M. The sign denotes if p lies within the mesh (negative) or outside the mesh (positive). Given the object mesh, M, in the palm frame, the hand joint configuration, Θ, and the environment as a set of objects, W, the truncated signed distance function can be written as:

$$C(\Theta, M, W) = \alpha_2 \sum_{w \in W} (\beta - \min(\beta, SD(M, w))) \quad (9)$$

which penalizes the object when it comes within β distance of any obstacles in the environment. Ideally collision functions should be used as constrains as in Schulman et al. (2014). However, we found that having the collision constraint as a cost term with a large scalar weight α_2 provided better trajectories and quicker solutions. Hence we add this as a cost term. This collision cost can be used to avoid both collisions with the environment and also with the hand. We add this as an additional cost term to Eq. 1 and perform trajectory optimization as before.

4.3 Object pose feedback controller

While our purely kinematic trajectory optimization performs well in practice, it still suffers some error during execution caused by friction, contact dynamics, and other unmodeled effects. Explicitly modeling these variables proves difficult and complex on real-world objects and impossibly to know prior to interaction with a novel object. As such we propose compensating for these errors through feedback controller based on visually tracking the object's pose. We use as targets the desired object pose trajectory $\mathbf{X}_D$ from initial pose X_0 to the desired object pose X_G generated by our "relaxed-rigidity" planner.

We define our object pose feedback controller to only affect the thumb joints, as we assume only the thumb attaches rigidly to the object during planning. To ensure the object remains in the robot's grasp, we track the planned joint trajectory Θ_D for the remaining fingers. As long as the object does not deviate from the planned trajectory by a large margin, thumb-only feedback should prove sufficient to maintain grasp of the object. (We validate this claim in Sect. 6.3.) The robot receives as input the joint position configuration $U[t]$ at every time step t. We define this as a combination of the feedforward planned joint trajectory $\Theta_D[t+1]$ and the object pose feedback term $\dot{\Theta}_{fb}$,

$$U[t] = \Theta_D[t+1] + \lambda_{fb}\dot{\Theta}_{fb}[t] \tag{10}$$

where the positive weight λ_{fb} allows for tuning the feedback compensation.

The feedback input $\dot{\Theta}_{fb}[t]$ corrects for errors between the planned fingertip pose and the predicted contact pose of the fingertip on the object. The planned fingertip pose at time step $t+1$ is given by $FK(\Theta_D[t+1])$. The predicted contact pose at time step $t+1$ is computed from the desired object pose $X_D[t+1]$ and the observed transformation matrix from fingertip to object frame, ${}^{O}\hat{T}_f$, as

$$H\left(X_D[t+1], {}^{O}\hat{T}_f\right) = Q\left(R(X_D[t+1]) \cdot {}^{O}\hat{T}_f\right) \tag{11}$$

where $R(\cdot)$ converts a pose into a homogenous transformation matrix and $Q(\cdot)$ transforms a homogenous matrix back into a pose. This essentially accounts for changes in rigid transformation between the object and the fingertip.

We define our feedback law by transforming the Cartesian space object pose error into the joint space using the inverse of the finger's Jacobian:

$$\dot{\Theta}_{fb}[t] = -J^{-1}_{\hat{\Theta}[t]}\left(FK(\Theta_D[t+1]) - H\left(X_D[t+1], {}^{O}\hat{T}_f\right)\right) \tag{12}$$

We found that approximating the Jacobian inverse by its transpose, rather than the Moore-Penrose pseudoinverse performed better for our underactuated fingers.

5 Implementation details and experimental protocol

We now describe important details relating to our implementation and the setup of our experiments.

5.1 Trajectory generation and feedback implementation

Direct methods for trajectory optimization, such as sequential quadratic programming (SQP), have shown promising results in robotics (Schulman et al. 2014; Posa and Cantu 2014; Posa et al. 2016). We solve our trajectory optimization problem using SNOPT (Gill et al. 2005) 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 i7-7700K CPU with 32 GB of RAM running Ubuntu 16.04 with ROS Kinetic (Quigley et al. 2009). The robot used is the Allegro Hand, which has four fingers with 4 joints each.[2] We solve for $T = 10$ time steps with each time step being $\Delta t = 0.167$ 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 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 ψ for the orientation cost term E_{or}.

The orientation cost E_{or} reduces orientation changes along the weight vector ψ. 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 $\psi = [0\ 1\ 0]$. We consider this weight as a trade off between allowing a larger manipulation workspace and enforcing smaller changes at contact points. We see in Sect. 6.1 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

[2] http://www.simlab.co.kr/Allegro-Hand.htm.

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

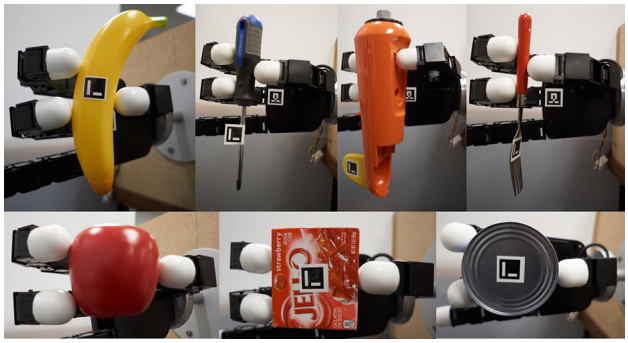


Fig. 4 Example grasps tested with various objects in our method

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 weights chosen for the extensions are $\alpha_1 = 0.01, \alpha_2 = 1000, \beta = 0.005, \lambda_{fb} = 50$.

For the collision avoidance experiments, the grasped object and the environment are approximately decomposed into convex groups using (Mamou and Ghorbel 2009) to speedup signed distance computation. We compute signed distances using libccd[3] based on a combination of the Gilbert–Johnson–Keerthi (GJK) algorithm and the expanding polytope algorithm (EPA), extensive details are found in (Van Den Bergen 2001). For object pose feedback controller, we use a GPU based particle tracker from Garcia Cifuentes et al. (2017) to track the object using a NVIDIA GTX 1060 GPU.

5.2 Experimental protocol

We selected objects of different size, texture and shape from the YCB dataset Calli et al. (2015), shown in Fig. 3, 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. Finding feasible desired poses given an initial grasp is a complex problem (Rojas and Dollar 2016; Hertkorn et al. 2013) 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. The Euclidean distance to the desired positions from the initial object positions, range from 0.8 to 8.33 cm with a mean of 4.87. Desired poses with small positional change have a large orientation change. One trajectory for each goal pose was generated.

The ground truth of the object pose is obtained using Aruco markers Garrido-Jurado et al. (2014). 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.

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 Sect. 2. The Allegro hand we use for physical validation has hemispherical fingertips which could cause rolling motion on the grasped object. Modeling rolling motion (Cutkosky and Wright 1986) between the fingertips and the grasped object requires extensive information about the object (surface geometry, friction) and precise force control of the fingertips. The Allegro hand's lack of joint level torque sensing prevents us from comparing our method to methods that use force control. We compare to the "point contact with friction" model which can

[3] https://github.com/danfis/libccd.

be approximated to a kinematic solution for object manipulation (Li et al. 1989). We formulate this as a trajectory optimization problem similar to our method with different cost terms. Specifically, we attempt to keep the fingertip positions fixed with respect to the object, while allowing the relative orientation to change. The cost function can be found in Sundaralingam and Hermans (2017).

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 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 (Huynh 2009).

$$err_{orient} = 100 \times \frac{\min(||q_d - q||_2, ||q_d + q||_2)}{\sqrt{2}} \tag{13}$$

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.

Ten unique reachable goal poses and two initial grasps per object are chosen to validate our planner. 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 0.59 cm with an associated variance of 0.09 cm. A total of 2000 trials are run across different methods to evaluate our proposed method.

To evaluate the joint acceleration extension to our planner and the object pose feedback controller, we conduct experiments with three objects—*Apple*, *Banana* and *Jello*. We choose 5 goals poses per object across two initial grasps per object. We run three trials per generated trajectory. To validate collision avoidance, we show two applications on a physical robot.

6 Results

We now discuss the results of our empirical experiments. We first validate our "Relaxed-Rigidity" planner on a real robot comparing with alternative formulations for in-grasp manipulation. We then discuss results from our extensions to the "Relaxed-Rigidity" planner. In all plots results correspond to objects grasped with three fingers, unless otherwise stated. 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.

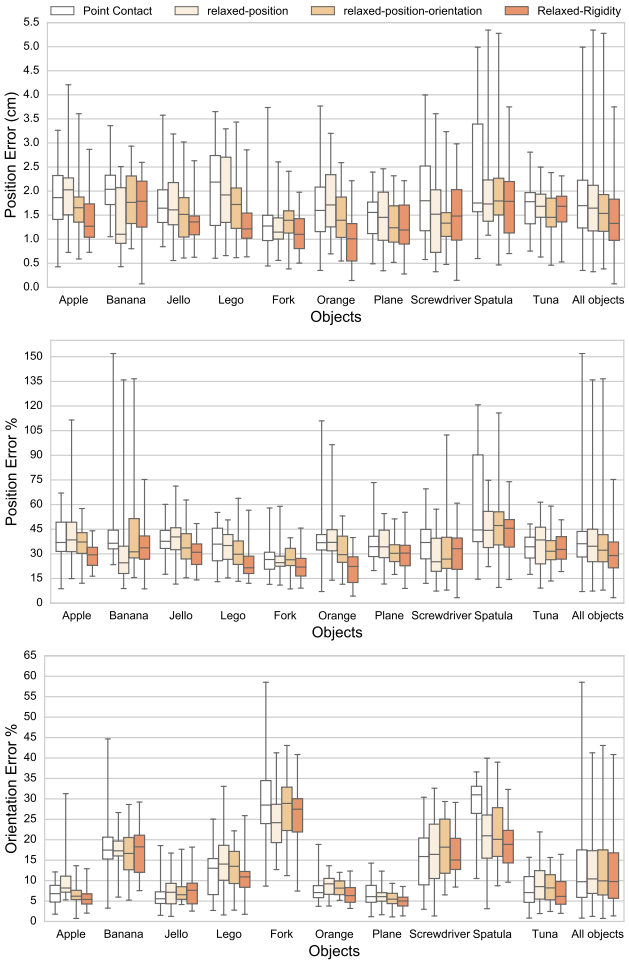


Fig. 5 A comparison of the relaxed-rigidity constraint performance with alternative formulations. Top: position error Middle: position error%. Bottom: orientation error%. The median position error decreases for all objects with our method. Except for *Banana* and *Jello*, the orientation error% improves for our method for all objects

6.1 Relaxed-rigidity physical robot validation

The position error and orientation error for all trials across all objects are shown in Fig. 5. 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 pose. Additionally, one can see that our method has a lower

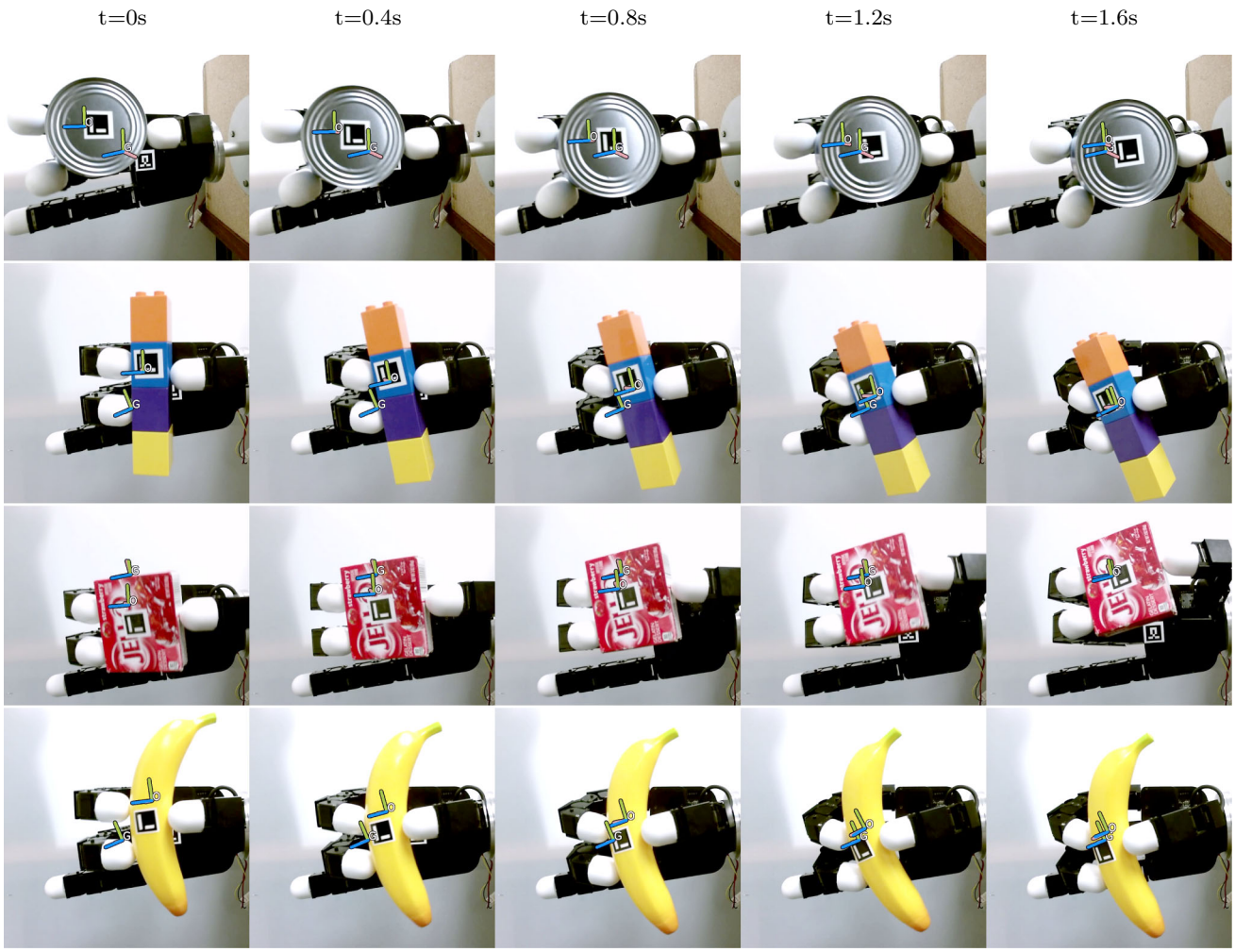


Fig. 6 Images showing manipulation of objects during trajectory execution. The trajectories are generated from our method ("Relaxed-Rigidity"). The frame 'O' represents the current object pose and 'G' frame is the desired object pose. *Tuna* being heavier than all other objects has a larger error due 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

variance in final position than the competing methods across nearly all objects. Four samples from our experiments are shown in Fig. 6 with overlaid current object pose and desired object pose.

Table 1 shows the success rate and the median errors across all these methods. The success rate and position error improve as we add additional costs from our method. It is also seen that our method performs better than assuming a point contact model. The point contact model also resulted in dropping the object on 25 out of 500 trials, while our proposed method never dropped an object. The orientation error for all methods remains low across all objects except for *Fork* where the fingertips are larger than the object causing it to roll with very small orientation changes at the fingertips. In all objects except *Banana* and *Jello*, the orientation error% improves with our method. A large improvement in orientation error is seen in *Spatula*, an object for which the point contact model with friction achieves relatively high orientation error.

To show our method generalizes to n-fingered grasps, we show results for 2-fingered and 4-fingered grasps in Fig. 7. We note that 2-fingered grasps tend to shake the object more during trajectory execution than 3-fingered grasps. With 4-fingered grasps, the ring finger sometimes loses and regains contact, adding little benefit over 3-fingered grasps.

6.2 Effect of joint acceleration

The inclusion of joint acceleration cost term, gives a smooth velocity profile for the object during the in-grasp manipulation as shown in Fig. 8. Linear interpolation has sudden jerks in the object trajectory if the goal pose is not reachable along

Table 1 Summary of results with the best value in bold text

Method	Suc.%	Pos. Error (cm)	Error% Pos.	Error% Orient.
PC	95	1.69	36.81	**9.74**
Relaxed-1	91	1.64	30.95	10.43
Relaxed-2	93	1.54	29.19	9.84
Relaxed-Rigidity	**100**	**1.32**	**28.67**	9.86

The errors are the median of all trials. "relaxed-1" refers to "relaxed-position" and "relaxed-2" refers to "relaxed-position-orientation"

the linear path as seen in Fig. 8. There was no significant difference in planning time and offline convergence errors between the two formulations. However, physical robot validation shows the the joint acceleration generates lower maximum position error and similar median position error to the linear interpolation, as shown in Fig. 9. The orientation error for *Banana* sees a significant improvement with "joint-acc" as it prevented rolling of the object during manipulation. The *Jello* object sees a significant reduction in position error as the smooth path reduces inertia caused by the powder moving inside the box. We infer the following from our validation: the exclusion of linear interpolation for the waypoint allows for finding a smooth trajectory to the desired pose, the smooth acceleration reduces rolling of the object due to rapid changes to object velocity, and Objects with non-rigidly attached parts have lower error as the smooth acceleration keeps inertia at a minimum.

6.3 Object pose feedback controller

We now show results for incorporating the object pose feedback controller on the original relaxed-rigidity planner with linear interpolation costs. Figure 9 shows that the feedback controller drastically reduced the variance in the position and orientation error. We note that nontrivial noise on the object pose persists, caused by the RGB-D based object tracker. This manifests by the lack of error reduction by the feedback controller when error is less than 1cm. Objects with an axis of symmetry such as the *Apple* object proved particularly difficult to track, since the particle filter was unable to find a unique pose.

6.4 Collision avoidance

An interesting application of in-grasp manipulation is to avoid collisions in a cluttered environment by making small changes to the object pose. We setup two such experiments and used our collision avoidance extension to generate trajectories. Figure 10 shows our in-grasp planner avoiding collisions with the environment while reaching the desired pose. This shows the effectivness of making small changes to

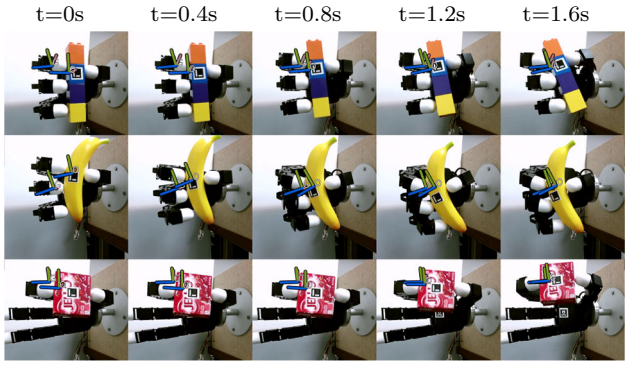


Fig. 7 Execution of in-grasp manipulation for four fingered and two fingered grasps. Frame "O" is the object pose and frame "G" is the goal pose. With the *Banana*, the ring finger loses contact during execution at t = 0.4 s but makes contact again at t = 0.8 s and the object reaches the desired pose

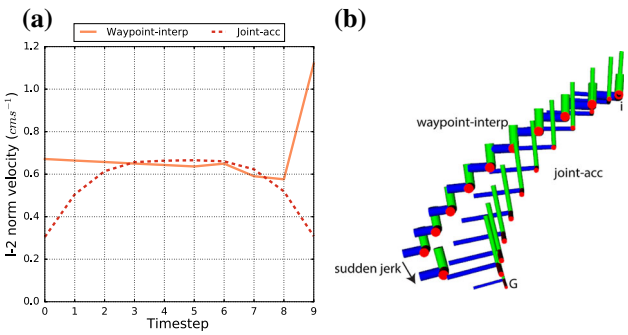


Fig. 8 **a** Shows the object velocity from the generated trajectory and **b** shows the object pose trajectory from "waypoint-interp" method as fat axes frames and "joint-acc" trajectory as slim tall axes frames. The waypoint interpolation method sees a sudden jump at timestep 8 which creates a jerk on the object as seen in **b**

the object pose to avoid obstacles in the environment which otherwise would require large motions with the arm. Adding the collision avoidance cost increased the planning time as we compute the signed distance in every iteration between the grasped object and the environment which we decompose into many convex obstacle. It took approximately 120 s to generate each collision free plan.

7 Discussion

We found several open questions to explore from extensive validation of our in-grasp manipulation planner which we discuss below.

7.1 Improving manipulation accuracy

Our planner was able to achieve an average position error of 13 mm without feedback of the object pose. While this might seem large, there are many tasks that could be performed with

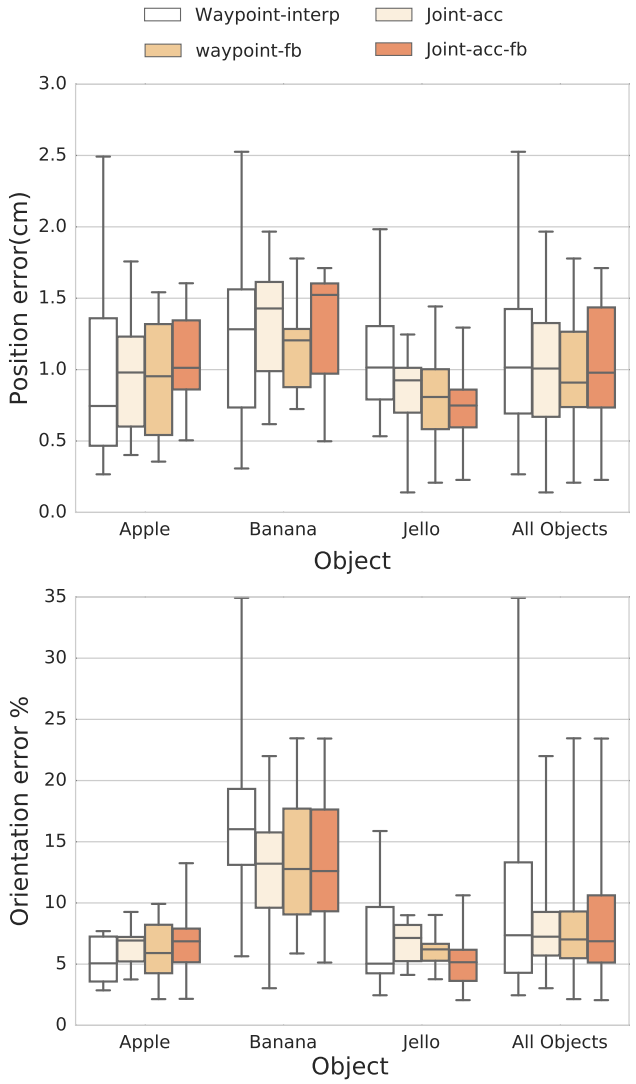


Fig. 9 Plots show the position and orientation error in reaching the desired pose across the four methods- "waypoint-interp" is our vanilla version, "joint-acc" is with our joint acceleration cost, and the methods with suffix "-fb" are run with the object pose feedback controller. Variance is reduced with the use of the feedback controller across all objects. Object *Jello* sees a significant reduction in position error due to reduced object dynamics excitation

this accuracy. One task we explore in this paper is moving a spoon into a cup (Fig. 10). If only the arm is used to move the spoon, a very precise arm controller or visual servoing is required to move inside the cup. With in-grasp manipulation, the spoon is moved inside the cup without visual servoing, using the dexterity in the fingers. At a broader scale, in-grasp manipulation cannot achieve large object pose changes, as the fingers have limited reachability. We have started exploring methods to switch to a different fingertip grasp to extend the reachable object poses (Sundaralingam and Hermans 2018). Two potential bottlenecks prevent us from improving the accuracy through online replanning: slow planning time and poor object pose tracking accuracy. Our current trajectory optimization implementation takes on an average 2 seconds to generate a trajectory. The optimization is computationally expensive as the reachability of the fingertips and the objective function are highly non-convex. This led us to use a Jacobian object pose feedback controller (Sect. 4.3). The feedback controller was unable to reduce the median object position error to less than 1 cm. Upon further analysis, we found the object pose tracker was not precise to less than 1 cm. We will explore improving the object pose tracking system and study the effect on manipulation accuracy. We will revaluate if the Jacobian controller is sufficient or online replanning is a necessity to improve accuracy.

7.2 Losing contact during manipulation

Physical experiments showed some of the fingers losing contact on the object during manipulation and making contact again before the manipulation is complete when four fingers were used as seen in Fig.7. This did not lead to dropping of the object. We will be exploring adding tactile feedback to maintain contact with the object. We never observed the object slipping from the grasp during manipulation.

7.3 Cost versus constraints

Our approach formulates the "relaxed-rigidity" terms as part of the cost as we want to minimize changes to the initial grasp as much as possible. Another perspective would be to formulate them as inequality constraints with thresholds (i.e. max allowed deviations). Formulating them as constraints provides a potential advantage of faster planning times. However, finding the thresholds for the "relaxed-rigidity" terms that would lead to successful executions on the physical robot is not straightforward. Additionally, a constraint based approach treats all feasible solutions equally while our approach attempts to minimize the deviation when possible.

8 Conclusion

We presented an 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. We implemented and experimentally validated the proposed method on a physical robot hand with ground truth error analysis. The results show that our relaxed-rigidity constraint allows better real-world performance than assuming a point contact model. We show how to use our planner with a collision avoidance cost to manipulate the grasped object in a cluttered environment. We show the ability to reduce unmodeled dynamic effects by adding a cost for smooth joint space paths. We show that use

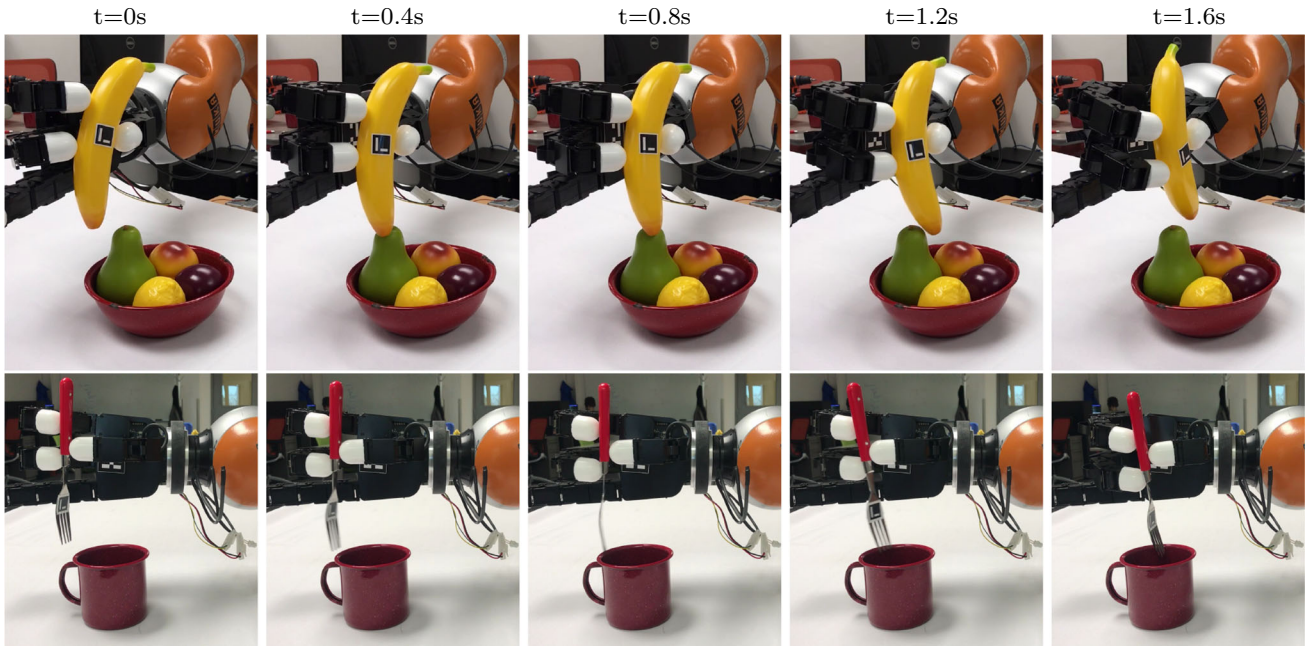


Fig. 10 Images showing collision-free manipulation of objects during trajectory execution. The *Banana* object moves around the pear fruit to avoid a collision and reaches its desired pose. The *Fork* object moves to inside the cup avoiding the brim of the cup

of an object pose feedback controller reduces the variance in trajectory execution.

Funding This study was funded partly by National Science Foundation (NSF) (Grant Number 1657596).

Compliance with ethical standards

Conflict of interest Author Balakumar Sundaralingam has no conflicts of interest. Author Tucker Hermans has no conflicts of interest.

Ethical approval This article does not contain any studies with human participants or animals performed by any of the authors.

References

Andrews, S., & Kry, P. G. (2013). Goal directed multi-finger manipulation: Control policies and analysis. *Computers & Graphics*, *37*(7), 830–839.

Bai, Y., Liu, & C. K. (2014). Dexterous manipulation using both palm and fingers. In *IEEE International Conference on Robotics and Automation (ICRA), IEEE* (pp. 1560–1565).

Bicchi, A. (2000). Hands for dexterous manipulation and robust grasping: A difficult road toward simplicity. *IEEE Transactions Robotics and Automation*, *16*(6), 652–662.

Bicchi, A., & Sorrentino, R. (1995). Dexterous manipulation through rolling. *IEEE International Conference on Robotics and Automation (ICRA), IEEE*, *1*, 452–457.

Calli, B., Singh, A., Walsman, A., Srinivasa, S., Abbeel, P., & Dollar, A. M. (2015). The YCB object and model set: Towards common benchmarks for manipulation research. In *International conference on advanced robotics (ICAR), IEEE* (pp. 510–517).

Carpin, S., Liu, S., Falco, J., & Van Wyk, K. (2016). Multi-fingered robotic grasping: A primer. arXiv preprint arXiv:160706620.

Ciocarlie, M., Goldfeder, C., & Allen, P. (2007). Dexterous grasping via eigengrasps: A low-dimensional approach to a high-complexity problem. In *Robotics: Science and systems manipulation workshop-sensing and adapting to the real world*, Citeseer.

Cutkosky, M. R., & Wright, P. K. (1986). Friction, stability and the design of robotic fingers. *Tohe International Journal of Robotics Research*, *5*(4), 20–37.

Fearing, R. S. (1986). Simplified grasping and manipulation with dextrous robot hands. *IEEE Transsctions Robotics and Automation*, *2*(4), 188–195.

Garcia Cifuentes, C., Issac, J., Wüthrich, M., Schaal, S., & Bohg, J. (2017). Probabilistic articulated real-time tracking for robot manipulation. *IEEE Robotics and Automation Letters (RA-L)*, *2*(2), 577–584.

Garrido-Jurado, S., Muñoz-Salinas, R., Madrid-Cuevas, F., & Marín-Jiménez, M. (2014). Automatic generation and detection of highly reliable fiducial markers under occlusion. *Pattern Recognition*, *47*(6), 2280–2292.

Gill, P. E., Murray, W., & Saunders, M. A. (2005). SNOPT: An SQP algorithm for large-scale constrained optimization. *SIAM Review*, *47*(1), 99–131.

Han, L., & Trinkle, J. C. (1998). Dextrous manipulation by rolling and finger gaiting. In *IEEE international conference on robotics and automation (ICRA), IEEE* (Vol. 1, pp. 730–735).

Han, L., Guan, Y. S., Li, Z., Shi, Q., & Trinkle, J. C. (1997). Dextrous manipulation with rolling contacts. In: *IEEE International conference on robotics and automation (ICRA), IEEE* (Vol. 2, pp. 992–997).

Hang, K., Li, M., Stork, J. A., Bekiroglu, Y., Pokorny, F. T., Billard, A., et al. (2016). Hierarchical fingertip space: A unified framework for grasp planning and in-hand grasp adaptation. *IEEE Transactions on Robotics*, *32*(4), 960–972.

Härtl, H. (1995). Dextrous manipulation with multifingered robot hands including rolling and slipping of the fingertips. *Robotics and Autonomous Systems*, *14*(1), 29–53.

Hertkorn, K., Roa, M. A., & Borst, C. (2013). Planning in-hand object manipulation with multifingered hands considering task constraints. In *IEEE International conference on robotics and automation (ICRA), IEEE* (pp. 617–624).

Hong, J., Lafferriere, G., Mishra, B., & Tan, X. (1990). Fine manipulation with multifinger hands. In *IEEE international conference on robotics and automation (ICRA), IEEE* (pp. 1568–1573).

Hoof, H. V., Hermans, T., Neumann, G., & Peters, J. (2015). Learning robot in-hand manipulation with tactile features. In *Proceedings IEEE/RAS international conference on humanoid robots (Humanoids)*.

Huynh, D. Q. (2009). Metrics for 3D rotations: Comparison and analysis. *Journal of Mathematical Imaging and Vision*, *35*(2), 155–164.

Kumar, V., Tassa, Y., Erez, T., & Todorov, E. (2014). Real-time behaviour synthesis for dynamic hand-manipulation. In *IEEE international conference on robotics and automation (ICRA), IEEE* (pp. 6808–6815).

Kumar, V., Todorov, E., & Levine, S. (2016). Optimal control with learned local models: Application to dexterous manipulation. In *IEEE international conference on robotics and automation (ICRA), IEEE* (pp. 378–383).

Li, Z., Hsu, P., & Sastry, S. (1989). Grasping and coordinated manipulation by a multifingered robot hand. *International Journal of Robotics Research*, *8*(4), 33–50.

Li, Q., Elbrechter, C., Haschke, R., & Ritter, H. (2013). Integrating vision, haptics and proprioception into a feedback controller for in-hand manipulation of unknown objects. In *2013 IEEE/RSJ international conference on intelligent robots and systems, IEEE* (pp 2466–2471).

Mamou, K., & Ghorbel, F. (2009). A simple and efficient approach for 3d mesh approximate convex decomposition. In 2009 16th IEEE international conference on image processing (ICIP), IEEE (pp. 3501–3504).

Mordatch, I., Popović, Z., & Todorov, E. (2012). Contact-invariant optimization for hand manipulation. In: *Proceedings of the ACM SIGGRAPH/Eurographics symposium on computer animation, Eurographics Association* (pp. 137–144).

Posa, M., Cantu, C., & Tedrake, R. (2014). Direct method for trajectory optimization of rigid bodies through contact. *International Journal of Robotics Research*, *33*(1), 69–81.

Posa, M., Kuindersma, S., & Tedrake, R. (2016). Optimization and stabilization of trajectories for constrained dynamical systems. In *IEEE international conference on robotics and automation (ICRA)*.

Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., Wheeler, R., & Ng, A. Y. (2009). Ros: an open-source robot operating system. In *ICRA workshop on open source software*, Kobe, Japan (vol 3, p. 5).

Rojas, N., & Dollar, A. M. (2016). Gross motion analysis of fingertip-based within-hand manipulation. *IEEE Transactions Robotics and Automation*, *32*(2), 1009–1016.

Rus, D. (1992). Dexterous rotations of polyhedra. In *IEEE international conference on robotics and automation (ICRA)*

Salisbury, J. K., & Craig, J. J. (1982). Articulated hands: Force control and kinematic issues. *The International journal of Robotics Research*, *1*(1), 4–17.

Salisbury, J. K., & Roth, B. (1983). Kinematic and force analysis of articulated mechanical hands. *Journal of Mechanisms, Transmissions, and Automation in Design*, *105*(1), 35–41.

Scarcia, U., Hertkorn, K., Melchiorri, C., Palli, G., & Wimböck, T. (2015). Local online planning of coordinated manipulation motion. In *IEEE international conference on robotics and automation (ICRA), IEEE* (pp. 6081–6087).

Schulman, J., Duan, Y., Ho, J., Lee, A., Awwal, I., Bradlow, H., et al. (2014). Motion planning with sequential convex optimization and convex collision checking. *Intl Journal of Robotics Research*, *33*(9), 1251–1270.

Srinivasa, S. S., Erdmann, M. A., Mason, M. T. (2005). Using projected dynamics to plan dynamic contact manipulation. In *IEEE/RSJ international conference on intelligent robots and systems (IROS), IEEE* (pp. 3618–3623).

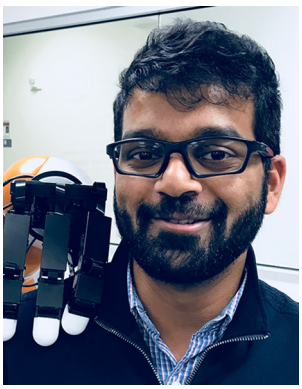
Sundaralingam, B., Hermans, T. (2017). Relaxed-rigidity constraints: In-grasp manipulation using purely kinematic trajectory optimization. In *Proceedings of robotics: Science and systems*, Cambridge, MA. https://doi.org/10.15607/RSS.2017.XIII.015.

Sundaralingam, B., & Hermans, T. (2018). Geometric in-hand regrasp planning: Alternating optimization of finger gaits and in-grasp manipulation. In IEEE international conference on robotics and automation (ICRA).

Toussaint, M. (2017). A tutorial on Newton methods for constrained trajectory optimization and relations to slam, Gaussian process smoothing, optimal control, and probabilistic inference. In *Geometric and numerical foundations of movements* (pp. 361–392). Berlin: Springer.

Van Den Bergen, G. (2001). Proximity queries and penetration depth computation on 3d game objects. In *Game developers conference* (Vol. 170).

Zucker, M., Ratliff, N., Dragan, a D, Pivtoraiko, M., Klingensmith, M., Dellin, C. M., et al. (2013). CHOMP: Covariant Hamiltonian optimization for motion planning. *The International Journal of Robotics Research*, *32*(9–10), 1164–1193. https://doi.org/10.1177/0278364913488805.



Balakumar Sundaralingam is a Ph.D. student at the University of Utah affiliated with the School of Computing and the Utah Robotics Center. He is advised by Dr. Tucker Hermans and works at the Learning lab for Manipualtion Autonomy. His current research focuses on in-hand manipulation and grasp planning using multi-fingered hands.

Tucker Hermans is an assistant professor in the School of Computing at the University of Utah, where he is affiliated with the Utah Robotics Center and is director of the Learning Lab for Manipulation Autonomy. Previously he was a postdoctoral researcher in the Intelligent Autonomous Systems lab at TU Darmstadt in Darmstadt, Germany. There he worked with Jan Peters on tactile manipulation and robot learning, while serving as the team leader at TUDa for the European Commission project TACMAN. He was at Georgia Tech from 2009 to 2014 in the School of Interactive Computing. There he earned his Ph.D. in Robotics under the supervision of Aaron Bobick and Jim Rehg in the Computational Perception Laboratory. His dissertation research dealt with robots learning to discover and manipulate previously unknown objects. At Georgia Tech he also earned a M.Sc. in Computer Science with specialization in Computational Perception and Robotics.