Object-Agnostic Dexterous Manipulation of Partially Constrained Trajectories

Andrew S. Morgan *, *Student Member, IEEE*, Kaiyu Hang *, *Member, IEEE*, and Aaron M. Dollar, *Senior Member, IEEE*

***Abstract*—We address the problem of controlling a partially constrained trajectory of the manipulation frame–an arbitrary frame of reference rigidly attached to the object–as the desired motion about this frame is often underdefined. This may be apparent, for example, when the task requires control only about the translational dimensions of the manipulation frame, with disregard to the rotational dimensions. This scenario complicates the computation of the grasp frame trajectory, as the mobility of the mechanism is likely limited due to the constraints imposed by the closed kinematic chain. In this letter, we address this problem by combining a learned, object-agnostic manipulation model of the gripper with Model Predictive Control (MPC). This combination facilitates an approach to simple vision-based control of robotic hands with generalized models, enabling a single manipulation model to extend to different task requirements. By tracking the hand-object configuration through vision, the proposed framework is able to accurately control the trajectory of the manipulation frame along translational, rotational, or mixed trajectories. We provide experiments quantifying the utility of this framework, analyzing its ability to control different objects over varied horizon lengths and optimization iterations, and finally, we implement the controller on a physical system.**

***Index Terms*—Dexterous manipulation, in-hand manipulation, manipulation planning.**

I. Introduction

DEXTEROUS manipulation is often characterized as the ability to reposition or reorient the object frame with respect to the hand frame [1]. Much work has addressed such an issue, providing generalized models that describe object frame trajectories given joint actuation velocities [2]. In many cases, however, the object frame is not necessarily the point on the object in which is desired to control. For example, in the task of handwriting, the position of the marker tip, which we denote as the manipulation frame, generally defines the precision of the inscribed character. In such a scenario, the *controlled dimensions* of the manipulation frame are purely translational, where we can largely relax the rotational constraint of the marker tip as to extend the task workspace. In other contexts, it may be required that purely rotational or even mixed trajectories are desired for task completion.

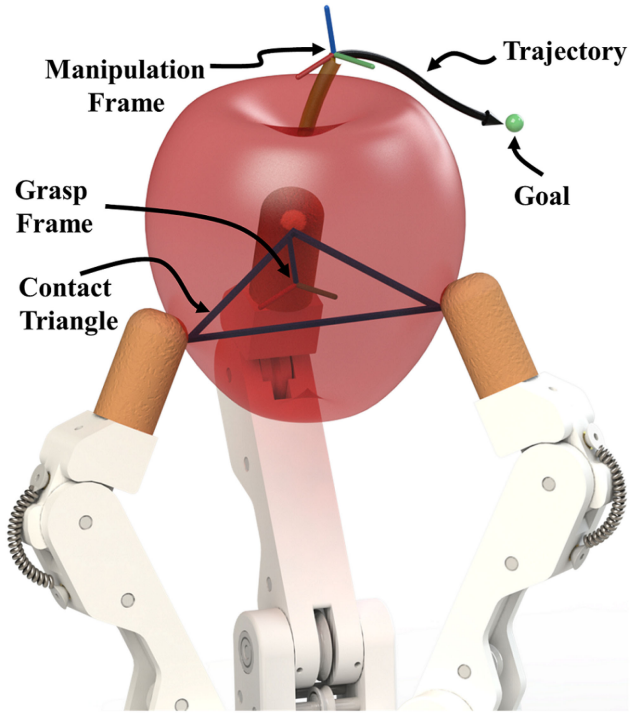


Fig. 1. Partially constrained trajectories of the manipulation frame, e.g. $\in \mathbb{R}^3$, leave uncertainties in grasp frame planning since the mobility of the mechanism is subject to constraints imposed by the closed kinematic chain. The proposed framework utilizes Model Predictive Control to solve for a valid grasp frame trajectory with any underconstrained reference.

In this letter, we build off the observation that many tasks require control about a partially constrained manipulation frame trajectory. In such cases, object (or grasp) frame trajectories in $SE(3)$ can be either difficult or impossible to analytically compute due to the absence of a one-to-one mapping, especially in an underactuated system where the hand's joint configuration is subject to both, kinematic and energy constraints. We propose an MPC-inspired control framework that utilizes an object-agnostic manipulation model and an energy-based propagation (or system dynamics) model of the hand. We differentiate between the *controlled dimensions* and the free dimensions of the manipulation frame, which can be any combination of dimensions in $SE(3)$.

Manuscript received February 24, 2020; accepted June 28, 2020. Date of publication July 7, 2020; date of current version July 21, 2020. This letter was recommended for publication by Associate Editor H. Liu and Mehmet R Dogar upon evaluation of the reviewers' comments. This work was supported by the U.S. National Science Foundation under Grant IIS-1734190. Andrew Morgan and Kaiyu Hang contributed equally to this work. *(Corresponding author: Andrew Morgan.)*

The authors are with the Department of Mechanical Engineering & Materials Science, Yale University, New Haven, CT 06520 USA (e-mail: andrew.morgan@yale.edu; kaiyuh@kth.se; aaron.dollar@yale.edu).

This article has supplementary downloadable material available at http://ieeexplore.ieee.org, provided by the authors.

Digital Object Identifier 10.1109/LRA.2020.3007467

Given a desired manipulation frame trajectory, a bidirectional initialization assumes the mobility of the hand is sufficient for the grasp frame to mimic the transformed trajectory for the next timestep, while leaving the free dimensions constant. By querying the learned model with this initialization, the resultant output is evaluated in a system propagation model. We repeat this process through a receding horizon to build the initial control trajectory. During this initialization, it is likely that the trajectory is inaccurate due to the limited mobility imposed on the mechanism by the closed kinematic chain. This issue is accounted for by optimizing grasp frame reference velocities in order to minimize trajectory error. We evaluate executions of various trajectories (translational, rotational, and mixed) with different control horizons and optimization iterations, and compare the results. In this work, we largely disregard object stability analyses due to the use of a compliant mechanism.

The contributions of this letter are twofold. First, we propose an optimization approach that extends the control capabilities of a generalized manipulation model, bypassing the need for task-specific training or modeling. Secondly, we underscore the advantage of using MPC for in-hand manipulation, which allows the system to recover from inaccurate system models or unmodeled contact scenarios.

II. Related Work

1) Analytical Modeling for Manipulation: Many works have approached dexterous manipulation with various levels of analytical modeling–from contact models [3] and fingerpad curvature models [4], to hand kinematic models [5] and whole hand-object system models [2]. Many powerful relationships have been formulated with such mathematical rigor. Although, the accuracy and efficacy of these models is highly subject to model parameters, which may be known *a priori* in structured settings, or may need to be estimated during manipulation via sensors on the hand, e.g. to leverage slip [6]. Some of these problems are nullified when using underactuated, adaptive hands that inherently reconfigure to uncertainties such as noisy control inputs or modeling errors [7]. Nevertheless, dexterous manipulation with such hands remains difficult to model as the output space is typically of higher dimension than the input space.

2) Learning for Manipulation: To overcome uncertainties in the analytical models, learning for manipulation–both model-based [8], [9] and model-free approaches [10]–has become popular as this approach is able to intrinsically estimate model parameters without user intervention. Consequentially, data for such approaches generally becomes too large to collect physically and must be done in simulation [11]. This caveat can be mitigated by relaxing the control dimensionality and constraints of the task, e.g. using a soft, compliant, or underactuated hand. While these hands are difficult to explicitly model, various works have introduced methods for closing the control loop through vision [12], [13] or through tactile sensing [14]. These works, however, focus mainly on the motion of the object/grasp frame and not on a generalized manipulation frame attached to the object.

3) Control for Manipulation: Control for manipulation has been similarly approached from various avenues–with methods based purely on kinematics [15], tactile sensing [16], and visual servoing [17], [18]. It is also possible to combine sensing modalities for additional control, e.g. for grasp adaptation [19]. However, each control approach is contingent on which sensing modalities are available. For example, underactuated hands are typically not equipped with joint encoders or tactile sensors, therefore, vision has become popular. In [20], joint configuration estimation was achieved through the use of particle filters and vision, therefore allowing more advanced control without the need for joint encoders. Regardless of these previous approaches, no works have embedded MPC with learning for controlling spatial trajectories with an underactuated hand.

III. Learning the Manipulation Model

In this section, we present an approach to learning the manipulation model of an underactuated hand through an energy-based perspective [12]. Throughout this letter, we assume all hand and object motions are quasistatic and the weights of the objects used are negligible–disregarding the need to explicitly model dynamics or object-specific properties, e.g. inertias. Moreover, we leverage a compliant end effector as these mechanisms are beneficial for maintaining stability of the hand-object system during manipulation, mitigating concerns of losing contact [7], [20].

A. The Grasp Frame

The establishment of the grasp frame generalizes the geometric properties of an arbitrary object within a grasp [21]. Fundamentally, it portrays the local geometry of the object and standardizes the representation of the object frame (Fig. 1, 2). We will reference the object frame as being one in the same as the grasp frame, as we expect object weights to be negligible. Assuming a single non-rolling contact is maintained on each fingertip of a hand with k fingers, let us define contact points $P = p_1, \ldots, p_k$ where $p_i \in \mathbb{R}^3, \forall i \in \{1, \ldots, k\}$ with respect to the hand frame. Noteworthily, with non-rolling contacts, any 3 points in P can explicitly define the grasp frame. For simplicity, let's assume p_1, p_2, and p_3 are used. Then, we can define the grasp frame pose, $\mathcal{X} \in SE(3)$, by Gram-Schmidt orthogonalization,

$$\begin{aligned} \mathcal{X} &= [\mathcal{G}_x, \mathcal{G}_y, \mathcal{G}_z | \mathcal{G}_o] \in SE(3) \\ \mathcal{G}_o &= \frac{1}{3}(p_1 + p_2 + p_3) \\ \mathcal{G}_x &= \frac{p_2 - p_1}{||p_2 - p_1||_2} \\ \mathcal{G}_z &= \frac{(p_3 - p_2) \times \mathcal{G}_x}{||(p_3 - p_2) \times \mathcal{G}_x||_2} \\ \mathcal{G}_y &= \mathcal{G}_z \times \mathcal{G}_x \end{aligned} \tag{1}$$

In this formulation, $\mathcal{G}_x, \mathcal{G}_y$, and $\mathcal{G}_z$ represent the directional vectors about the x, y, and z axes, respectively, with reference to the origin, $\mathcal{G}_o$. Using the same object contact points, we can calculate the contact triangle relationship,

$$\mathcal{T} = (||p_1 - p_2||_2, ||p_2 - p_3||_2, ||p_3 - p_1||_2) \in \mathbb{R}^3 \tag{2}$$

representing the distance between fingertips in contact with the object, where $\mathcal{T} = (\mathcal{T}_1, \mathcal{T}_2, \mathcal{T}_3)$. It is important to note that this formulation generalizes object geometry but not necessarily

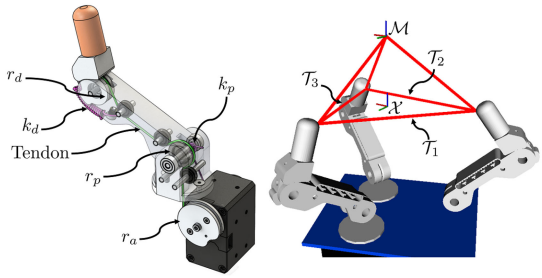


Fig. 2. Left: The tendon transmission of an underactuated finger is dependent on pulley and spring parameters. Right: Object geometry can be generalized by evaluating the triangle relationship, $\mathcal{T}$, between the contacts, and offsetting the manipulation frame, $\mathcal{M}$, from the grasp frame, $\mathcal{X}$.

object dynamics. Additional generalization of object dynamics will be addressed in future work.

B. Learning From the Energy Model

Underactuated systems can be modeled in terms of energy, where the joint configuration, $q \in \mathbb{R}^{\sum_{i=1}^{k} j_i}$, of a hand that has j_i joints per finger, equilibrates such that the internal energy of the system is minimized. We represent the actuation position as a, where $dim(a) < dim(q)$ in an underactuated system. Given an actuation velocity, $\dot{a}$, and the grasp frame, $\mathcal{X}_t$, at time t, the energy-based propagation model (or system dynamics model) provides a prediction for the next step of the grasp frame pose, $\mathcal{X}_{t+1}$. This transition is calculated given a tendon transmission constraint,

$$r_{ai}\dot{a}_i = r_{pi}\dot{q}_{pi} + r_{di}\dot{q}_{di} \tag{3}$$

and the contact triangle constraint, $\mathcal{T}_t = \mathcal{T}_{t+1}$. Thus, we can find the equilibrated joint configuration of the hand, q^* by,

$$q^* = argmin \sum_i E^i(q^i) \;\; \text{s.t.} \;\; (2), (3) \tag{4}$$

where E^i is the potential energy of the ith finger,

$$E^i(q^i) = \frac{1}{2}(k_p q_{pi}^2 + k_d q_{di}^2) \tag{5}$$

Here, r_{pi}, r_{di}, and r_{ai} are the radii of the pulleys on the proximal joint, distal joint, and actuator, respectively, on finger i (Fig. 2). Similarly, $\dot{q_{pi}}, \dot{q_{di}}$, and $\dot{a}_i$ are the rotational velocities about the same joint on the same finger.

This energy-based propagation model enables efficient data collection in simulation, and has shown to easily transfer to a physical system [12]. By predefining various contact relationships in $\mathcal{T}$ and applying a random actuation input, $\dot{a}$, we observe the grasp frame transition from $\mathcal{X}_t$ to $\mathcal{X}_{t+1}$, thus calculating $\dot{\mathcal{X}} \in se(3)$ by taking the element-wise difference. With a 15-dimensional input feature, $s_n = (\mathcal{X}_n, \dot{\mathcal{X}}_n, \mathcal{T}_n)$, and an output feature, $\dot{a}_n$, we build the training set,

$$\mathcal{S} = \{s_n\}_{n=1:N}, \;\; \mathcal{R} = \{\dot{a}_n\}_{n=1:N}$$

where N denotes training sample size. With these action-reaction pairs, we create a Random Forest Regression model,

$$g : (\mathcal{X}, \dot{\mathcal{X}}, \mathcal{T}) \rightarrow \dot{a} \tag{6}$$

that maps the current pose of the grasp frame, the desired grasp frame velocity, and the contact triangle relationship to an actuation velocity. This learned model will be further utilized in the proposed control framework.

IV. Control Framework

For the continuation of this work, the main control process is illustrated in Fig. 3 and is notated as follows:

- t denotes the current time and $t+n$ denotes n steps into the future (e.g. $\mathcal{M}_{t+3}$ is the predicted manipulation frame $\in SE(3)$ in three timesteps)
- dotted variables represent the change from t, one timestep forward (e.g. $\dot{\mathcal{X}} = [\mathcal{X}_t - \mathcal{X}_{t+1}] \in se(3)$)
- barred variables represent the initialization guess during the $bidirInit(\cdot)$ process, which has not yet been executed by the propagation model (e.g. $\bar{\mathcal{X}}_{t+1} \in SE(3)$)
- primed variables have been executed by the propagation model and are the resultant configuration after $(iter)$ optimization iterations (e.g. $\mathcal{M}'_{t+3}(25)$ if $iter = 25$)

A. Model Predictive Control

The proposed control framework utilizes Model Predictive Control (MPC) with an optimizer based on Stochastic Hill Climbing as to extend the task workspace. MPC is advantageous for manipulation, as the next control input is optimized after each system step. This property helps mitigate error caused by inaccurate propagation models or when unmodeled contact scenarios occur, e.g. rolling or slip.

MPC evaluates the cost of an input over a user defined prediction/control horizon, k_p. This horizon dictates how far in advance the controller evaluates its trajectory, while maintaining integrity on any system constraints, e.g. actuation constraints or energy constraints. In this work, we seek to control a subset of the manipulation frame's dimensions (referenced as the *controlled dimensions*) while allowing the free dimensions to move as to satisfy the system constraints. The manipulation frame, $\mathcal{M} \in SE(3)$, is a frame of reference rigidly attached to the grasp frame, $\mathcal{X}$, which would typically be affixed to a feature on the object. Let's define our desired reference trajectory as r, comprised of m waypoints in the *controlled dimensions*. We can define the *controlled dimension* set as $c \subset (x, y, z, \theta_R, \theta_P, \theta_Y)$, which can be any combination of translational and rotational components for a desired trajectory. We denote the *controlled dimensions* of the manipulation frame as $\mathcal{M}_c$.

While accounting for kinematic, energy, and actuation constraints, we seek to minimize the error between $\mathcal{M}_{c,t}$ and $r[w_t]$, where w_t is the waypoint on r currently closest to $\mathcal{M}_{c,t}$. Additionally, we impose an extra penalty on how far $\mathcal{M}_{c,t}$ is from the goal position, r_{end}. We therefore formulate the cost function J,

$$J = \sum_{i=1}^{k_c} \gamma ||r[w_{t+i}] - \mathcal{M}_{c,t+i}||_2 + \ldots \\ \sigma ||r_{end} - \mathcal{M}_{c,t+i}||_2 + \lambda ||\dot{a}_{t+1}||_2 \tag{7}$$

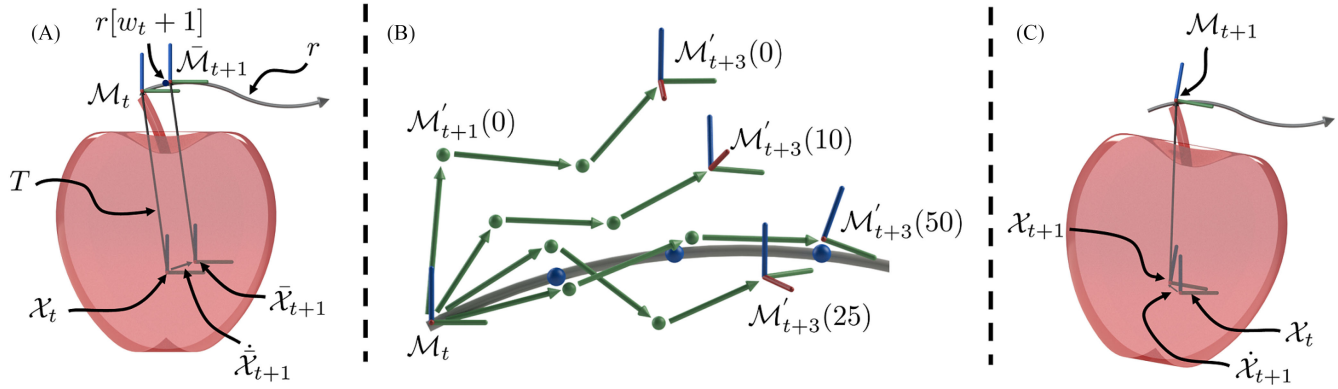


Fig. 3. A.) The manipulation frame, $\mathcal{M}_t$, can be represented by a rigid transformation, T, from the grasp frame, $\mathcal{X}_t$. In Algorithm 2 a bidirectional guess initializes the model's input variables by assuming that the next grasp frame pose, $\bar{\mathcal{X}}_{t+1}$, has the same velocity, $\dot{\bar{\mathcal{X}}}_{t+1}$, as the underconstrained manipulation frame trajectory transitioning $\mathcal{M}_t$ to $\bar{\mathcal{M}}_{t+1}$, which is located on the next trajectory waypoint $r_m[w_t+1]$. B.) While this bidirectional guess serves well for initialization, kinematic and energy constraints likely limit mobility and may not allow the grasp frame to move desirably. Thus, the resultant pose evaluated in the propagation model, $\mathcal{M}'_{t+1}(0)$, does not follow the path. The optimization then perturbs the grasp frame velocities of the best trajectory $iter$ times and evaluates the result in propagation model. This depicts a trajectory convergence with a horizon $k_p = 3$. C.) After optimization, the first actuation input of the best evaluated trajectory is executed, providing our true next grasp frame pose $\mathcal{X}_{t+1}$ and our next manipulation frame pose $\mathcal{M}_{t+1}$.

where γ, σ, and λ are weightings that are tuned heuristically to penalize the trajectory error, trajectory length, and the actuation input, respectively. In tuning, for example, if it is desired to increase execution speed, increasing σ and decreasing γ and λ will do this with the trade-off of likely decreasing trajectory accuracy.

B. The Manipulation Controller

Using this cost-minimization approach, we formulate the control process as illustrated in Fig. 3 and as outlined in Algorithm 1. We attempt to optimize a controlled trajectory, $\mathcal{C}_i$, to closely follow r. These controlled trajectories are constructed with a chain of $k_p + 1$ nodes, where k_p is the prediction horizon. Each node is referenced in the trajectory chain with zero-based indexing, so, $\mathcal{C}_i.n[2]$ is the third node. Each node has 3 properties–the current grasp frame ($\mathcal{X}$), the grasp frame velocity input evaluated in the previous node ($\dot{\mathcal{X}}$), and the actuation velocity used by the propagation model in the previous node ($\dot{a}$). Each $\mathcal{C}_i$ therefore has a cost defined by (7) that can be used to compare the utility of each trajectory.

1) Initializing the Trajectory: Given r, which has the same dimensionality as c–that can be any combination of dimensions in $SE(3)$–the control process begins by constructing the initial trajectory, $\mathcal{C}_{best}$. This process is outlined in lines 1-11 of Algorithm 1 and is depicted in Fig. 3.A.

To formulate the first trajectory, we rely on a bidirectional initialization presented in Algorithm 2. This procedure initializes a first guess for the grasp frame velocity, $\dot{\bar{\mathcal{X}}}_{t+1}$, by assuming that the kinematic constraints of the hand allow for identical movement about the grasp frame as that of the manipulation frame. This process begins by computing the closest waypoint, $r[w_t]$, from $\mathcal{M}_t$ to the reference trajectory. We make a guess that the manipulation frame would like to move to the next waypoint $r[w_t+1]$ while attempting to keep the free dimensions constant. Through this notion, we calculate a guess for the next state of the manipulation frame, $\bar{\mathcal{M}}_{t+1} \in SE(3)$. A transformation, T, can then be computed relating $\mathcal{X}_t$ to $\mathcal{M}_t$. This process becomes bidirectional as we apply the inverse of T to $\bar{\mathcal{M}}_{t+1}$ to obtain a guess for the next state of the grasp frame, $\bar{\mathcal{X}}_{t+1}$. The grasp frame velocity guess, $\dot{\bar{\mathcal{X}}}_{t+1}$, is finally estimated by taking the element-wise difference between $\mathcal{X}_t$ and $\bar{\mathcal{X}}_{t+1}$.

After the bidirectional initialization guess, $\dot{\bar{\mathcal{X}}}_{t+1}$ is evaluated in the learned model $g(\cdot)$, given the current pose of the node. This resultant actuation velocity, $\dot{a}_{t+1}$, is executed in the propagation model, providing the next state grasp frame pose, $\mathcal{X}'_{t+1}(0)$. The true grasp frame velocity, $\dot{\mathcal{X}}'_{t+1}(0)$ is then calculated by taking the difference between $\mathcal{X}_t$ and $\mathcal{X}'_{t+1}(0)$. These variables are then added to the trajectory, $\mathcal{C}_{best}$ and the entire process is repeated over the entire length of the control horizon, or until the distance between the manipulation frame and the endpoint of the trajectory is less than a threshold, ϵ.

2) Trajectory Optimization: Once the first trajectory is generated, initialized as $\mathcal{C}_{best}$, we construct $iter$ temporary trajectories that attempt to reduce the cost as defined by (7). Here, $iter$ represents the number of optimization iterations we intend to compute. This process is depicted in Fig. 3.B and references lines 13-25 of Algorithm 1.

Given the grasp frame velocity of the node in timestep $(t+1)$ of the best trajectory, $\mathcal{C}_{best}$, we perturb its value with a normal distribution of predefined interval limits. This result, $\dot{\mathcal{X}}'_{t+1}(i)$, where i is the current value of $iter$, is calculated in $perturb(\cdot)$–Stochastic Hill Climbing's exploration method (Algorithm 3). The learned model then evaluates this grasp velocity to form the actuation velocity, $\dot{a}_{t+1}$. We execute $\dot{a}_{t+1}$ in the propagation model to determine the next grasp frame state $\mathcal{X}'_{t+1}(i)$ at optimization iteration i. The resultant node is then added to $\mathcal{C}_i$ and the process continues over the entire prediction horizon. If the manipulation frame is found to have reached within some distance threshold, ϵ, the loop breaks prematurely. Once a trajectory of $k_p + 1$ in length is computed, we compare

Algorithm 1: MPC with Stochastic Hill Climbing Optimization.

Input: $\mathcal{X}_t$, r, c, k_p, $\mathcal{T}$, $iter$, ϵ
Output: $\dot{a}$

$\mathcal{C}_{best} \leftarrow Trajectory()$ ▷ initialize first trajectory
$\mathcal{C}_{best}.addNode(\mathcal{X}_t, \dot{\mathcal{X}}_0 = 0, \dot{a}_0 = 0)$ ▷ start node
for $t = 1$ to k_p **do** ▷ prediction horizon
  $\bar{\dot{\mathcal{X}}}_{t+1} \leftarrow bidirInit(\mathcal{C}_{best}.n[t].\mathcal{X}, r, c)$ ▷ Algorithm 2
  $\dot{a}_{t+1} \leftarrow g : (\mathcal{C}_{best}.n[t].\mathcal{X}, \bar{\dot{\mathcal{X}}}_{t+1}, \mathcal{T})$ ▷ (6)
  $\mathcal{X}'_{t+1}(0) \leftarrow Hand.evaluate(\dot{a}_{t+1})$ ▷ (4)
  $\dot{\mathcal{X}}'_{t+1}(0) \leftarrow diff(\mathcal{C}_{best}.n[t].\mathcal{X}, \mathcal{X}'_{t+1}(0))$
  $\mathcal{C}_{best}.addNode(\mathcal{X}'_{t+1}(0), \dot{\mathcal{X}}'_{t+1}(0), \dot{a}_{t+1})$
  $\mathcal{M}_{t+1}(0) \leftarrow Hand.manipFrame(\mathcal{X}'_{t+1})$
  if $||\mathcal{M}_{c,t+1}(0) - r_{end}||_2 < \epsilon$ **then**
    break ▷ reached goal

for $i = 1$ to $iter$ **do** ▷ optimization iterations
  $\mathcal{C}_i \leftarrow Trajectory()$ ▷ initialize new trajectory
  $\mathcal{C}_i.addNode(\mathcal{X}_t, \dot{\mathcal{X}}_0 = 0, \dot{a}_0 = 0)$
  for $t = 1$ to k_p **do**
    $\dot{\mathcal{X}}'_{t+1}(i) \leftarrow perturb(\mathcal{C}_{best}.n[t+1].\dot{\mathcal{X}})$ ▷ Algorithm 3
    $\dot{a}_{t+1} \leftarrow g : (\mathcal{C}_i.n[t].\mathcal{X}, \dot{\mathcal{X}}'_{t+1}(i), \mathcal{T})$ ▷ (6)
    $\mathcal{X}'_{t+1}(i) \leftarrow Hand.evaluate(\dot{a}_{t+1})$ ▷ (4)
    $\mathcal{C}_i.addNode(\mathcal{X}'_{t+1}(i), \dot{\mathcal{X}}'_{t+1}(i), \dot{a}_{t+1})$
    $\mathcal{M}_{t+1}(i) \leftarrow Hand.manipFrame(\mathcal{X}'_{t+1}(i))$
    if $||\mathcal{M}_{c,t+1}(i) - r_{end}||_2 < \epsilon$ **then**
      break ▷ reached goal
  if $Cost(\mathcal{C}_i) < Cost(\mathcal{C}_{best})$ **then** ▷ (7)
    $\mathcal{C}_{best} = \mathcal{C}_i$ ▷ better trajectory

return $\mathcal{C}_{best}.n[1].\dot{a}$

Algorithm 2: $bidirInit(\cdot)$.

Input: $\mathcal{X}_t$, r, c
Output: $\bar{\dot{\mathcal{X}}}_{t+1}$

$\mathcal{M}_t \leftarrow Hand.manipFrame(\mathcal{X}_t)$
$w_t \leftarrow nearestWaypoint(\mathcal{M}_{c,t}, r)$
for l in $[x, y, z, \theta_R, \theta_P, \theta_Y]$ **do**
  if $l \subset c$ **then**
    $\bar{\mathcal{M}}_{l,t+1} \leftarrow r[l, w_t + 1]$
  else
    $\bar{\mathcal{M}}_{l,t+1} \leftarrow \mathcal{M}_{l,t}$
$T \leftarrow getTransform(\mathcal{X}_t, \mathcal{M}_t)$
$\bar{\mathcal{X}}_{t+1} \leftarrow applyInvTransform(\bar{\mathcal{M}}_{t+1}, T)$
$\bar{\dot{\mathcal{X}}}_{t+1} \leftarrow diff(\mathcal{X}_t, \bar{\mathcal{X}}_{t+1})$
return $\bar{\dot{\mathcal{X}}}_{t+1}$

the costs of the best trajectory, $\mathcal{C}_{best}$, with the cost of the current trajectory, $\mathcal{C}_i$. If this cost is smaller, we replace $\mathcal{C}_{best}$ with $\mathcal{C}_i$ and continue this loop until the number of desired iterations is satisfied.

The algorithm concludes by returning the first actuation input of the best trajectory, $\mathcal{C}_{best}.n[1].\dot{a}$. This input is then executed physically (Fig. 3.C) and results in the actual system transition from $\mathcal{M}_t$ to $\mathcal{M}_{t+1}$, and similarly, $\mathcal{X}_t$ to $\mathcal{X}_{t+1}$. Algorithm 1 is repeated until the trajectory goal is reached.

It is important to note that the algorithm does not require that each waypoint in r is passed through, as it may be the case that some points along the trajectory are infeasible given the constraints of the system. To account for this, only the initialization step attempts to follow a waypoint, while the optimization steps minimize the trajectory cost by staying within a close distance and extending towards the end goal.

Algorithm 3: $perturb(\cdot)$

Input: $\dot{\mathcal{X}}_t$
Output: $\dot{\mathcal{X}}'_{t+1}$

$\delta_x, \delta_y, \delta_z \leftarrow translationalLimit$
$\delta_{\theta_R}, \delta_{\theta_P}, \delta_{\theta_Y} \leftarrow rotationalLimit$
for i in $[x, y, z, \theta_R, \theta_P, \theta_Y]$ **do**
  $\dot{\mathcal{X}}'_{t+1} \leftarrow \dot{\mathcal{X}}_t + rand.uniform(-\delta_i, \delta_i)$
return $\dot{\mathcal{X}}'_{t+1}$

V. Experimentation

The proposed control framework was instantiated on a 3-fingered underactuated Yale Openhand Model O. Physical modifications to the readily available open source design include a rounded fingertip and pulleys/bearings within the finger as to reduce friction in the tendon's transmission. Each finger, composed of two links, is actuated by a single Dynamixel XM-430 motor with return forces supplied by springs at each of the joints (Fig. 2).

The learned model in (6) was trained with a dataset of size 300,000 over 50 different contact triangles, $\mathcal{T}$, by evaluating the input-output relationship after random actuation of the energy model in (4). A Random Forest model of tree depth 10 and forest size of 30 was trained, which accounted for joint limits and actuation constraints. Due to the different values in $\mathcal{T}$ used for training, the learned model was able to generalize over different object geometries, which is beneficial as it enables adaption to undesired contact scenarios where the relational geometry between the fingertips change, e.g. rolling or slip, as previously presented in [12].

A. Translational Trajectory Control

We implemented translational control, i.e. $c = (x, y, z)$, in a simulated environment (Fig. 2) while varying the control horizon and number of optimization iterations as to tune the controller. This test, presented in Fig. 4, tracks the x, y, z position of the manipulation frame over time in an attempt to trace the letters 'GRABLAB'. Depicted in different colors, three different-sized objects were used in experimentation, with properties presented in Fig. 5. Each letter was 20 mm in height and 10 mm in width and was written within the $x - y$ plane. Letters were comprised of a number of goal points–squares (start), circles (intermediate), and stars (end)–with 50 waypoints in between each goal.

Fig. 4(A) depicts a test correlating accuracy to varying horizon lengths and optimization iterations. Generally, we note that as

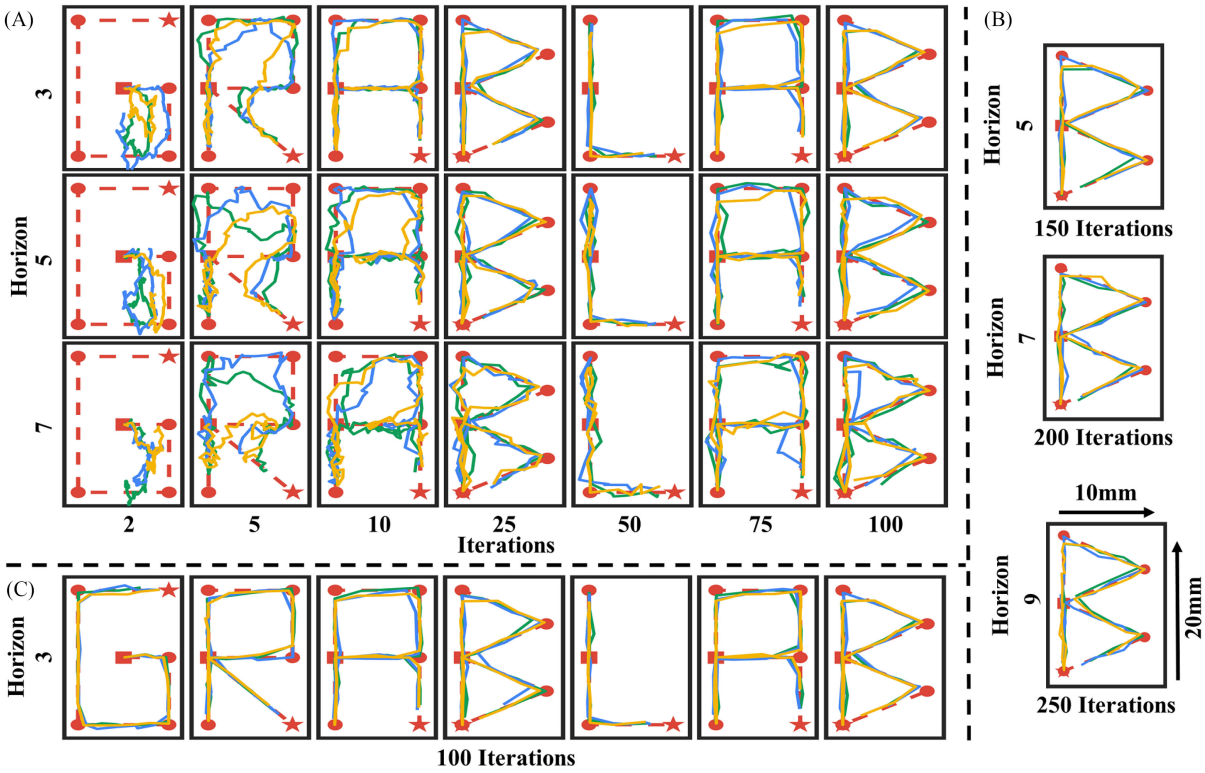


Fig. 4. Translation control, $c = (x, y, z)$, of the manipulation frame depicting the reference trajectory in the $x - y$ plane (Red), and the trajectories of Obj. 1 (Green), Obj. 2 (Yellow), and Obj. 3 (Blue). A.) We trace the letters 'GRABLAB' while varying control horizons and optimization iteration lengths. As we increase the number of iterations, the manipulation frame trajectory becomes more accurate. We see that with fewer iterations, the manipulation frame is not able to follow the desired trajectory. B.) When the control horizon increases, subsequently, the number of optimization iterations must as well to realize similar trajectories. C.) Tracing the word 'GRABLAB' with the most precise control horizon/iteration pair (horizon of 3 and 100 iterations).

Obj. #	$\mathcal{T}_1$ (mm)	$\mathcal{T}_2$ (mm)	$\mathcal{T}_3$ (mm)	T_p (mm)
1	98.1	81.3	108.5	(0, 0, 50)
2	73.2	59.7	78.6	(-20, 0, 40)
3	65.2	59.1	71.2	(0, 15, 60)

Fig. 5. Properties for the three objects used in simulation. The transformation, T, assumes that the manipulation frame, $\mathcal{M}$, and the grasp frame, $\mathcal{X}$, have the same orientation, but are offset by the positional vector T_p.

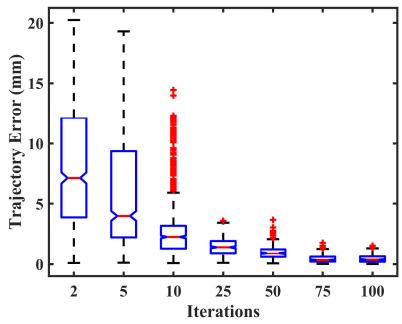


Fig. 6. With a prediction horizon of 3, the letters 'GRABLAB' were traced with three different objects while varying optimization iterations. The error experienced during execution was recorded for each of the trajectories. We identify an elbow point of 50 iterations satisfies the desired task accuracy.

the number of iterations increases (horizontal axis), the accuracy of the manipulation frame trajectory similarly increases. We note that it is likely that more iterations are needed for longer control horizons. This observation is evaluated in Fig. 4(B), where we record similar trajectory errors (0.72 mm mean) while increasing the number of iterations for longer horizons (5, 7, and 9). We then present the best recorded accuracy for the tracing of 'GRABLAB' in Fig. 4(C), with a horizon of 3 and 100 iterations.

Quantitatively, we tune the control parameters by evaluating manipulation frame trajectory accuracy while fixing the horizon length to 3 and altering the number of optimization iterations. We note that in the task of scripting, a trajectory error of less than 2 mm is sufficient for legibility. Testing up to 100 iterations (0.45 mm error), the results show that 50 iterations (0.95 mm error) is sufficient to satisfy the accuracy required by the task, presented in Fig. 6. For this reason, we will proceed in the next sections by evaluating trajectories with this configuration.

B. Rotational and Mixed Trajectory Control

In addition to a purely translational trajectory about the manipulation frame, we test the control approach with other partially constrained trajectories, namely, a purely rotational trajectory $c = (\theta_R, \theta_P, \theta_Y)$, and a mixed trajectory, $c = (z, \theta_R, \theta_Y)$. This choice of trajectories further underscores the diversity of dimensional combinations which can be inherently accounted for in this framework, after retuning weighting parameters in the cost function and scaling the *controlled dimensions* to characteristic length.

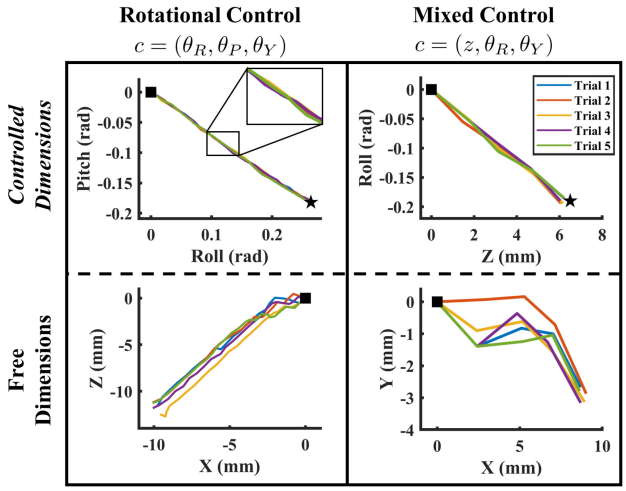


Fig. 7. A single trajectory in Rotation Control (left) and a single trajectory in Mixed Control (right) was executed for 5 trials. The *controlled dimensions* (top) follow the trajectory as desired. The free dimensions (bottom) are allowed to drift to any trajectory that adheres to the system constraints. The start configuration is denoted with a square and the goal configuration (only in the *controlled dimensions*) is denoted with a star.

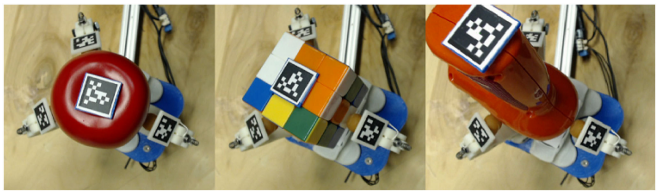


Fig. 8. Top view of the apple, Rubik's Cube, and drill from the YCB Object and Model Set used for physical testing of the control framework.

In each of these tests, the hand was initialized with the same hand configuration as in Fig. 2, using Obj. 1. With a horizon of 3 and with 50 optimization iterations, a goal trajectory was formed transitioning $\mathcal{M}$ from its current state to a goal configuration. Five trials were executed, resetting the hand after each trial. We record the state of the manipulation frame along the execution trajectory. As presented in Fig. 7, the trajectory of $\mathcal{M}$ was able to successfully follow the desired control trajectory ($0.52 \pm 0.3^\circ$ error for rotations). During this execution, we illustrate how the free dimensions are able to drift so long as system constraints are satisfied, and thus do not need to follow the same trajectory each trial. This concept is depicted in the bottom of the figure, where we note a trajectory deviation between trials.

C. Physical Translation Control

We employed the devised control framework on a physical system as to complete the tracing of letters 'RAL' with three different objects from the YCB Object and Modeling Set (Objs. #23, 72, 77) [22]. In this case, we employed translational *controlled dimensions*, $c = (x, y)$, scripting in the plane orthogonal to the palm as to maintain readability of the completed manipulation. The three objects, depicted in Fig. 8, were tracked by affixing 6-D pose AprilTags to the object, serving as the manipulation frame. The pose of the marker was then tracked by

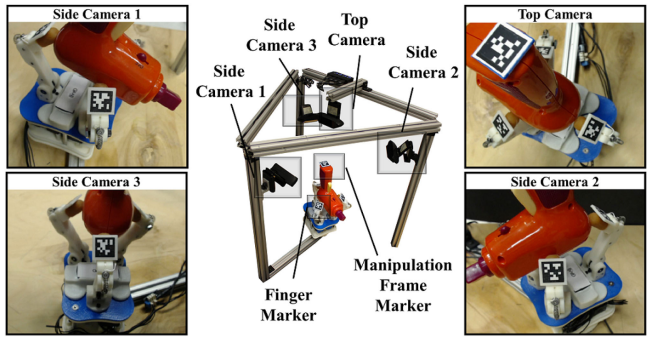


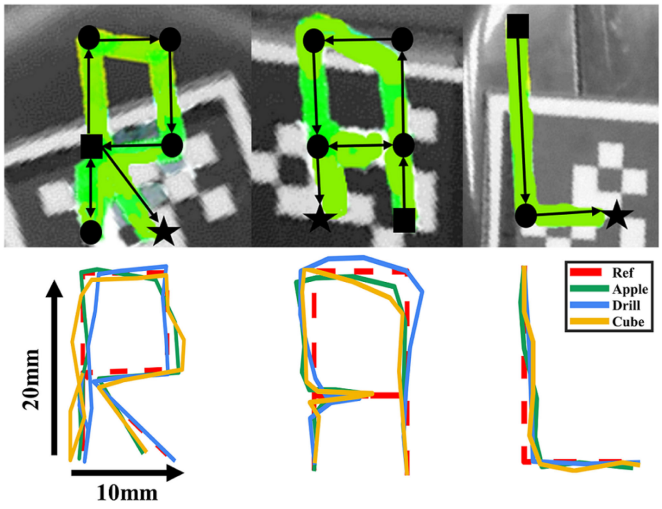
Fig. 9. A 4-camera tracking system records both, the pose of the grasp frame and the pose of the manipulation frame via attached markers.

Obj.	$\mathcal{T}_1$ (mm)	$\mathcal{T}_2$ (mm)	$\mathcal{T}_3$ (mm)	T_p (mm)
Apple	67.9	57.4	65.7	(3.5, 5.1, 48.6)
Drill	64.9	50.1	66.2	(5.9, -8.2, 121.2)
Cube	63.6	57.1	64.1	(-2.3, -4.2, 37.9)

Fig. 10. Grasp and transformation properties of the apple, drill, and Rubik's Cube used in physical experimentation. T_p is the translational offset of the grasp frame to the manipulation frame in x, y, z directions.

an overhead camera. The control framework relies on knowing the current configuration of the hand in order to compute the next actuation input, therefore, we placed 3 additional cameras around the hand–developing a 4-camera setup that is able to track the configuration of each finger in addition to the configuration of the object (Fig. 9). Markers were placed on the back of each fingertip and a transformation from the finger markers computes the contact location, and thus the pose of the grasp frame.

The markers were affixed to each object as follows: placed on the stem of the apple, placed on the bottom of the handle of the drill, and placed on the top (any) surface of the Rubik's Cube (Fig. 8). This generated initial contact triangle relationships and transformations from the grasp frame to the manipulation frame as presented in Fig. 10. We employed a prediction horizon of 3 and set $iter$ to 50. As presented in Fig. 11, each letter was comprised of a set of goal points, which constructed a system of trajectories approximately 20 mm in height and 10 mm in width. The task started with the center of the manipulation frame marker in the square starting position. At this point, a new trajectory was formed with 50 waypoints providing the path from the current start location to the first goal point. After the actuation input was solved through the MPC framework, the hand executed the result and evaluated how close it was to the goal point. If the manipulation frame was within a 2 mm threshold, a new trajectory was formed and the manipulation frame would attempt to move towards the next goal point until completion. During this process and after each input execution, the grasp frame $\mathcal{X}$, the manipulation frame $\mathcal{M}$, and the contact triangle relationship $\mathcal{T}$ were updated as to account for any undesired rolling or sliding of the contacts. Each letter was traced with the three aforementioned YCB objects and the execution times and average trajectory errors were recorded. We noted that the greatest error was when tracing of the letter 'A,' but was only slightly higher than the letter 'R'. This is likely attributed to



Letter	R	A	L
Goal Points	8	7	3
Avg. Time (s)	82.4	91.3	32.4
Avg. Err. (mm)	1.23±0.37	1.42±0.45	0.53±0.24

Fig. 11. The letters 'RAL' were traced with the manipulation frame on a physical system for 3 different objects ($k_p = 3$, $iter = 50$). Top: Three example executions of writing the letters R (traced with the apple), A (traced with the Rubik's Cube), and L (traced with the drill) are presented with their associated goal points. Middle: The path following accuracy for all three objects tracing letters 'RAL'. Bottom: The average time and trajectory errors recorded during execution for all three objects.

the cross-bar tracing that stopped prematurely. Since we did not greatly penalize the input actuation velocity, i.e. λ was small, we noted large motions in physical execution, typically requiring 2-3 actuation sequences to reach from goal point to goal point. Overall, these executions resulted in clear, discernible capitalized characters of 'RAL'.

VI. Discussions and Future Work

In this letter, we addressed the problem controlling partially constrained trajectories about the manipulation frame based on a planning-enabled MPC framework. This work extends the utility of generalized manipulation models as it is a way to better satisfy trajectory requirements of various tasks. We tested this approach by constraining different dimensions of the trajectory–translational, rotational, and mixed–and we showed that the controller was able to accurately follow the *controlled dimensions* while allowing the free dimensions to drift. We found that, generally, a horizon length of 3 with 50 iterations was sufficient for convergence that satisfied our task requirements. This may not be the case, however, in more complex tasks that typically operate at the boundary of system constraints. In such cases, more sophisticated parameter tuning and extension of the prediction horizon may be necessary for a smooth transition to a valid configuration.

In future work, we are interested in further defining this framework for maintaining hand-object stability–which was largely disregarded in this work since mechanism compliance generally provided stable grasps. Additional accuracy is also likely possible while accounting for the mass-related dynamics of the hand and of the object. By incorporating such components, we believe this framework will be extremely valuable for extending robot manipulation capabilities.

References

[1] A. Bicchi, "Hands for dexterous manipulation and robust grasping: A difficult road toward simplicity," *IEEE Trans. Robot. Autom.*, vol. 16, no. 6, pp. 652–662, Dec. 2000.
[2] R. M. Murray, S. S. Sastry, and L. Zexiang, *A Mathematical Introduction to Robotic Manipulation*. Boca Raton, FL, USA: CRC Press, Inc., 1994.
[3] J. C. Trinkle, J.-S. Pang, S. Sudarsky, and G. Lo, "On dynamic multi-rigid-body contact problems with coulomb friction," *J. Appl. Math. Mechan.*, vol. 77, no. 4, pp. 267–279, 1997.
[4] D. J. Montana, "Contact stability for two-fingered grasps," *IEEE Trans. Robot. Autom.*, vol. 8, no. 4, pp. 421–430, Aug. 1992.
[5] T. Okada, "Computer control of multijointed finger system for precise object-handling," *IEEE Trans. Syst., Man, Cybernet.*, vol. SMC-12, no. 3, pp. 289–299, May 1982.
[6] D. L. Brock, "Enhancing the dexterity of a robot hand using controlled slip," in *Proc. IEEE Int. Conf. Robot. Autom.*, 1988, pp. 249–251.
[7] A. M. Dollar and R. D. Howe, "The highly adaptive SDM hand: Design and performance evaluation," *Int. J. Robot. Res.*, vol. 29, no. 5, pp. 585–597, Apr. 2010.
[8] N. Fazeli, S. Zapolsky, E. Drumwright, and A. Rodriguez, "Learning data-efficient rigid-body contact models: Case study of planar impact," in *Proc. 1st Annu. Conf. Robot Learn.* (Proc. Mach. Learn. Res.), vol. 78. S. Levine, V. Vanhoucke, and K. Goldberg, Eds. PMLR, Nov. 2017, pp. 388–397. [Online]. Available: http://proceedings.mlr.press/v78/fazeli17a/fazeli17a.pdf
[9] A. Gupta, C. Eppner, S. Levine, and P. Abbeel, "Learning dexterous manipulation for a soft robotic hand from human demonstrations," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2016, pp. 3786–3793.
[10] A. Rajeswaran *et al.*, "Learning complex dexterous manipulation with deep reinforcement learning and demonstrations," in *Proc. Robot.: Sci. Syst.*, Pittsburgh, Pennsylvania, Jun. 2018, doi: 10.15607/RSS.2018.XIV.049.
[11] M. Andrychowicz *et al.*, "Learning dexterous in-hand manipulation," *Int. J. Robot. Res.*, vol. 39, no. 1, pp. 3–20, 2020.
[12] A. S. Morgan, K. Hang, W. G. Bircher, and A. M. Dollar, "A data-driven framework for learning dexterous manipulation of unknown objects," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2019, pp. 8273–8280.
[13] A. Sintov, A. S. Morgan, A. Kimmel, A. M. Dollar, K. E. Bekris, and A. Boularias, "Learning a state transition model of an underactuated adaptive hand," *IEEE Robot. Autom. Lett.*, vol. 4, no. 2, pp. 1287–1294, Apr. 2019.
[14] H. van Hoof, T. Hermans, G. Neumann, and J. Peters, "Learning robot in-hand manipulation with tactile features," in *Proc. IEEE-RAS Int. Conf. Humanoid Robots*, 2015, pp. 121–127.
[15] B. Sundaralingam and T. Hermans, "Relaxed-rigidity constraints: Kinematic trajectory optimization and collision avoidance for in-grasp manipulation," *Auton. Robots*, vol. 43, no. 2, pp. 469–483, 2019.
[16] B. Ward-Cherrier, N. Rojas, and N. F. Lepora, "Model-free precise in-hand manipulation with a 3d-printed tactile gripper," *IEEE Robot. Autom. Lett.*, vol. 2, no. 4, pp. 2056–2063, Oct. 2017.
[17] B. Calli and A. M. Dollar, "Robust precision manipulation with simple process models using visual servoing techniques with disturbance rejection," *IEEE Trans. Autom. Sci. Eng.*, vol. 16, no. 1, pp. 406–419, Jan. 2019.
[18] N. Furukawa, A. Namiki, S. Taku, and M. Ishikawa, "Dynamic regrasping using a high-speed multifingered hand and a high-speed vision system," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2006, pp. 181–187.
[19] K. Hang *et al.*, "Hierarchical fingertip space: A unified framework for grasp planning and in-hand grasp adaptation," *IEEE Trans. Robot.*, vol. 32, no. 4, pp. 960–972, Aug. 2016.
[20] K. Hang, W. G. Bircher, A. S. Morgan, and A. M. Dollar, "Hand–object configuration estimation using particle filters for dexterous in-hand manipulation," *Int. J. Robot. Res.*, 2019. [Online]. Available: https://journals.sagepub.com/eprint/KRX4NV4CGVKZI5XAUEP7/full
[21] K. Tahara, S. Arimoto, and M. Yoshida, "Dynamic object manipulation using a virtual frame by a triple soft-fingered robotic hand," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2010, pp. 4322–4327.
[22] B. Calli *et al.*, "Yale-cmu-berkeley dataset for robotic manipulation research," *Int. J. Robot. Res.*, vol. 36, no. 3, pp. 261–268, 2017.