

## Abstract

### Observing, Controlling, and Planning Compliant Robot Manipulation

Andrew S. Morgan

2023

*Complex robots complicate control; keep designs simple and exploit emergent behavior.*

In-hand manipulation is a complicated process—it requires a group of individual manipulators to work in proper unison with one another, modulating forces on an object while making and breaking contact. Traditional approaches to realizing such capabilities have been through the use of overly-complex, anthropomorphic, and thus expensive, robot hands equipped with a vast array of sensors, which are often noisy and can in turn lead to task failure. Mechanically compliant end effectors, on the other hand, have shown to be beneficial to robot manipulation, particularly for grasping, as they are able to “absorb the slack” in any modeling, sensing, or control uncertainty. As grasping is a necessary element to in-hand manipulation, it can further be hypothesized that compliant end effectors could be similarly beneficial for extending these capabilities. However, motions of compliant mechanisms are typically complex and difficult to analytically model. Moreover, there remain questions in how to appropriately determine and represent system state—what features need to be tracked, how accurate do they need to be, and how often do they need to be captured? These questions can be studied through the lens of grasp mechanics via vision-based feedback, which can play a crucial role for such devices that often lack onboard sensing as to keep designs simple, compact, and inexpensive. Formally, I will present an approach that is able to observe the state of a compliant hand-object system during manipulation, control the object along a planned path for fixed-contact in-hand manipulation, and finally, desirably plan the trajectory of a grasped object within-hand given constraints. To close, I will extensively showcase the applicability of these methods in tight tolerance and open-world assembly tasks.

Observing, Controlling, and Planning Compliant Robot Manipulation

A Dissertation  
Presented to the Faculty of the Graduate School  
of  
Yale University  
in Candidacy for the Degree of  
Doctor of Philosophy

by

Andrew S. Morgan

Dissertation Director: Prof. Aaron Dollar

May, 2023



Copyright © 2023 by Andrew S. Morgan  
All rights reserved.

To my grandparents

— the ones who taught me to shoot for the moon but only settle for what lies beyond —

“This grumpy old racecar I know once told me somethin’. It’s just an empty cup.”

-Lightning McQueen

# Contents

<b>Acknowledgements</b>	<b>xx</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Motivation for Society . . . . .	1
1.2 The Human Hand . . . . .	2
1.3 Research Overview . . . . .	3
1.4 Thesis Overview . . . . .	4
<b>2 Observing the State of a Complaint Hand</b>	<b>6</b>
2.1 Introduction . . . . .	6
2.2 Related Work . . . . .	10
2.3 Grasp Mechanics-Based Features . . . . .	13
2.3.1 Finger Manipulability Measures . . . . .	13
2.3.2 Grasp Quality Measures . . . . .	16
2.3.3 Hand-Object Manipulability Measures . . . . .	17
2.3.4 Curvature of Contact . . . . .	18
2.4 Bounding Feature Generalizability . . . . .	19
2.4.1 Mechanics of Underactuated Manipulation . . . . .	20
2.4.2 Mode Characterization in Simulation . . . . .	22
2.4.3 Bounding Feature Distributions by Statistical Testing . . . . .	24
2.5 Self-Supervised Tagging and Object Reset . . . . .	26
2.5.1 Manipulation Primitives . . . . .	26
2.5.2 Geometric Hand-Object Representation . . . . .	27

2.5.3	Self-Supervised Mode Detection . . . . .	28
2.5.4	Standardizing Object Reset . . . . .	30
2.6	Data Collection . . . . .	33
2.7	Experiments . . . . .	37
2.7.1	Classifier Identification and Observation Reduction . . . . .	37
2.7.2	Classification Accuracy . . . . .	38
2.7.3	Feature Reduction . . . . .	39
2.7.4	Single-Component Feature Reduction . . . . .	43
2.7.5	Online Classification . . . . .	43
2.8	Discussions . . . . .	45
<b>3</b>	<b>Controlling a Compliant Hand for In-hand Manipulation</b>	<b>47</b>
3.1	Introduction . . . . .	47
3.2	Related Work . . . . .	49
3.3	Devising a Manipulation Model . . . . .	50
3.3.1	The Grasp Frame . . . . .	50
3.3.2	Learning from the Energy Model . . . . .	51
3.4	Controlling In-Hand Manipulation . . . . .	54
3.4.1	Model Predictive Control . . . . .	55
3.4.2	The Manipulation Controller . . . . .	56
3.5	Experiments . . . . .	60
3.5.1	Translational Trajectory Control . . . . .	60
3.5.2	Rotational and Mixed Trajectory Control . . . . .	63
3.5.3	Physical Translation Control . . . . .	65
3.6	Discussions . . . . .	68
<b>4</b>	<b>Planning Finger Gaiting for Compliant Hands</b>	<b>69</b>
4.1	Introduction . . . . .	69
4.2	Related Work . . . . .	72
4.3	Preliminaries . . . . .	73
4.3.1	Constrained Within-Mode Planning . . . . .	73

4.3.2	Safe Mode Transitions . . . . .	74
4.4	Orientation-based Motion Planning . . . . .	76
4.4.1	Proper Rotations in $SO(3)$ . . . . .	76
4.4.2	Planning Orientation Transitions . . . . .	78
4.5	Object Control . . . . .	80
4.5.1	Translational Planning . . . . .	80
4.5.2	Orientation and Translation Control . . . . .	81
4.5.3	Generalization and Practical Algorithm Modifications . . . . .	81
4.6	Experimental Setup . . . . .	83
4.6.1	Robot Setup . . . . .	83
4.6.2	Mode Design . . . . .	85
4.7	Experiments . . . . .	87
4.7.1	Safe Modes Characterization . . . . .	87
4.7.2	Single Trajectory Execution . . . . .	87
4.7.3	Continuous Goal Trajectories and Robustness . . . . .	88
4.7.4	Generalization to Object Geometry . . . . .	89
4.8	Discussions . . . . .	90
<b>5</b>	<b>Robot Assembly with Compliant Robots</b>	<b>92</b>
5.1	Introduction . . . . .	92
5.2	Vision-based Tight Tolerance Insertion . . . . .	93
5.2.1	Introduction . . . . .	93
5.2.2	Related Work . . . . .	95
5.2.3	Proposed Framework . . . . .	97
5.2.4	Insertion Strategy . . . . .	102
5.2.5	Experiments . . . . .	108
5.2.6	Discussions . . . . .	115
5.3	Force-based Tight Tolerance Insertion . . . . .	117
5.3.1	Introduction . . . . .	117
5.3.2	Related Work . . . . .	120

5.3.3	Methodology . . . . .	122
5.3.4	Experiments . . . . .	126
5.3.5	Discussions . . . . .	132
<b>6</b>	<b>Conclusion</b>	<b>133</b>
6.1	Summary . . . . .	133
6.2	Lessons Learned and Future Work . . . . .	134
6.3	Suggestions for Continuing this Line of Work . . . . .	135
<b>A</b>	<b>Notes on Visual-based Insertion</b>	<b>137</b>

# List of Figures

2.1	Geometric features can be extracted visually during manipulation with a priori knowledge of the fingertip geometry, object geometry, and the number of finger links. (Top) A pivot-flexure finger manipulates a pear-shaped object with rolling contacts (Mode: Normal). (Bottom) A three-link pivot finger manipulates a rectangular object until sliding occurs along the left finger (Mode: Sliding). . . . .	7
2.2	By predicting modes of manipulation before or at the moment they occur, the user is able to transition between modes (trigger or avoid) for desired manipulation. Modes are typically detected when the hand-object system is in a similar configuration as those shown. . . . .	8
2.3	Free swing manipulability workspaces for both proposed manipulability measures. (Top) Free swing trajectories for three two-link planar fingers used in this work. (Bottom) Free swing trajectory of the three-link planar finger used in this work. Unlike the penalized manipulability measure, the standard manipulability measure does not account for joint hard stops. . . . .	15
2.4	(A) Annotation of hand parameters required for modeling underactuated manipulation mechanics. (B) Proximal and distal link lengths in simulation are changed by the same value, $\Delta l$ , as to maintain unit length during simulation.	21



2.5	MANOVA mode distribution testing of grasp mechanics-based features and position-based features. Green cells generally indicate the extended bounds by which mechanics-based features are able to transfer beyond that of their position-based counterpart (blue cells). Red cells indicate that neither of the feature sets likely share data distributions with the base variant (solid black cell). . . . .	25
2.6	Simple manipulation primitives enable planar motion within the workspace of a Yale OpenHand Model T42 gripper. These primitives enable the object to move up, down, left, or right depending on the Cartesian velocity reference, $\mathbf{v}$ . . . . .	27
2.7	Sliding contacts are detected by verifying rolling contact constraints cannot be satisfied. In this depiction, we track the velocity of the hand-object contact points and ensure they are within some normed threshold between each other.	31
2.8	An object crane and stabilization beam with affixed magnets accurately resets the object into the same configuration for each trial. This allows us to sequentially collect large amounts of data for training. . . . .	32
2.9	Manipulation was performed on 6 different gripper variants. The base variant used in training, the symmetric PL-PL gripper, was evaluated with four different objects (small circle, large circle, small rectangle, and large rectangle). A total of 3500 points for training were collected for the four identified modes. The five test variants (PL-PS, PS-PL, PL-PLsq, PS-FL, and PS-PS-PS) then performed manipulation with two of the six test objects. Two novel objects were added in testing (medium oval and medium pear). During manipulation, 50 occurrences of each mode were collected for each gripper-object combination. A quarter is placed next to the objects for size reference. <i>*Sliding only occurs with rectangular objects, therefore limiting the number of sliding cases.</i> . . . . .	34
2.10	Six objects were used for testing and training. In the manipulation plane, object geometries are classified either as a circle, rectangle, oval, or pear. .	35

2.11	Depicting regions of the workspace where modes typically occur. Markers indicate the centroid of the object when a mode was detected. We note the symmetry of this illustration, were modes typically occur mirrored across the central axis of the gripper. . . . .	35
2.12	Validation of training data size by reducing observations. . . . .	38
2.13	(Left three columns) Confusion matrices for each gripper variant given differing feature sets (described in Sec. 2.7.3). (Right column) Object centroid position for modes detected within the workspace of each gripper variant. (Light Blue-Drop, Dark Blue-Normal, Yellow-Stuck, Red-Sliding) . . . . .	40
2.14	Feature importance measures provided by the Random Forest Classifier via Gini impurity. Features in blue are included in Feature Sets 1,2,3, features in red are included in Feature Sets 1,2, and features in green are included in Feature Set 1. See Table 2.4. . . . .	41
2.15	Five-fold cross validation accuracy of the PL-PL training variant. Features were reduced one at a time subject to their classification accuracy contribution (see Fig. 2.14). . . . .	44
2.16	Online classification of two novel gripper variants. The arrow signifies the Cartesian velocity reference and the text (Drop or Sliding) signifies the predicted mode. (Left) A PS-FL left finger and a PS-PS-PS right finger perform manipulation and the online classifier predicts a drop will occur given the Cartesian velocity reference. (Right) A PS-FL left finger and a PL-PS right finger predicts sliding will occur during manipulation. . . . .	45
3.1	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. . . . .	52

3.2	(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, $\mathcal{X}$ . . . . .	53
3.3	A.) The manipulation frame, $\mathcal{M}_t$ , can be represented by a rigid transformation, $T$ , from the grasp frame, $\mathcal{X}_t$ . In Alg. 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 <i>iter</i> 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 $\mathcal{M}_{t+1}$ . . . . .	55
3.4	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). . . . .	61

3.5	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. . . . .	63
3.6	A single trajectory in Rotation Control (left) and a single trajectory in Mixed Control (right) was executed for 5 trials. The <i>controlled dimensions</i> (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 <i>controlled dimensions</i> ) is denoted with a star. . . . .	64
3.7	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. . . . .	65
3.8	A 4-camera tracking system records both, the pose of the grasp frame and the pose of the manipulation frame via attached markers. . . . .	66
3.9	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. . . . .	67
4.1	We explore the development of a complete $SO(3)$ planner for within-hand manipulation using finger gaits, by controlling two orthogonal extrinsic rotation axes. Given a start configuration (cube face A), the proposed planner finds an action sequence along the two controlled dimensions so as to reach the desired goal configuration (cube face B). During manipulation, the pose of the object is tracked via a low-latency, 6D pose object tracker, providing feedback for online replanning and disturbance compensation via a recovery phase that uses translation control. . . . .	70

4.2	Multi-modal planning problems can be conceptualized as operating in different configuration manifolds. (a) Given a single manifold $\mathcal{M}$ , the planner must find a path along the constrained layer. (b) A single mode, or foliation, can have multiple manipulation manifolds depending on the start configuration of the hand-object system. (c) Switching modes is possible when the system finds a configuration, $q'$ , that lies on both manifolds. (d) Finding a path from start configuration, $q_s$ , to goal configuration, $q_g$ , can require multiple jumps between modes. Note that although represented in this figure as folds, we assume modes to be differentiable but not necessarily euclidean. . . . .	74
4.3	(a) Our planning solution for $SO(3)$ begins by enumerating outward the goal orientation manifold by step size $\sigma_R$ and creating a KD-Tree. (b) After creation, a forward search outward is performed from the start orientation, $R_s$ , combining a $z$ -axis rotation first, then an $x$ -axis rotation. (c) This search continues until a candidate orientation is within some distance $\rho$ from the expanded goal manifold. (d) Finally, the entire plan is enumerated from start to goal along the acquired trajectory with appropriate timestamps and step size $\sigma_R$ . . . . .	76
4.4	(a) Our system is comprised of a RGBD camera fixated to the robot's environment. (b) During manipulation, the pose of the object, e.g., bunny, is tracked via a 6D pose object tracker. (c) A bottom view of the Yale Model Q illustrates the $110^\circ$ abduction capabilities of the differentially coupled fingers. . . . .	83
4.5	The Yale OpenHand Model Q is capable of operating along four modes for fingertip-based manipulation. Modes are shown with transitions from the start configuration (center). . . . .	84
4.6	(a) We experimentally validate our method with 5 different objects, namely a cube, sphere, toy truck, Stanford Bunny, and toy duck. (b) Tessellated faces of the cube show poses of the affixed letters. . . . .	85

4.7	The safety of mode transitions are attributed to the reconfigurability of the underactuated mechanism. (a) Starting rotations around different axes of the object elicit different amounts of reconfiguration upon regrasping. (b) This is particularly apparent in the yaw ( $z$ -axis) direction of the object, where the object follows the minimum energy configuration of the mechanism, and hence allowing us to estimate $\rho$ from Eq. (4.3). . . . .	86
4.8	We execute a single planned trajectory 8 times and record its repeatability. (a) Controlled dimensions of the object trajectory, such as roll and yaw, follow closely in all trials, where as uncontrolled dimensions, such as $x$ -axis translation and pitch, are allowed to drift. (b) Modes are enacted at different times in the manipulation, including recovery phases when the object is not in the center of the graspable workspace. . . . .	88
4.9	We test system robustness by (a) following through an extended trajectory of cube faces F, A, B, C, D, E, and F once again, and (b) deliberately applying different perturbations to the object along controlled dimensions so as to require online replanning and recovery. . . . .	88
4.10	Our manipulation planner is able to extend to objects of convex and non-convex geometries, and reach any orientation in $SO(3)$ . . . . .	89
5.1	(a) An RGBD-based 6D object pose tracker monitors the task state, serving as the primary sensing modality for the robot performing a variety of insertion tasks with tight tolerances, (b) the sequence of cup stacking. . . . .	93
5.2	System pipeline: (red) a visual tracking framework trained solely with synthetic data to estimate 6D pose differences, provides feedback for (blue) manipulation planning and control of a low-impedance manipulator and a compliant end-effector performing within-hand manipulation for insertion tasks. . . . .	98
5.3	Physics-aware, high-fidelity synthetic training data are augmented via domain randomization. . . . .	100
5.4	6D pose tracking on RGBD image observations streamed from the camera. . . . .	100

5.5	(a) Object geometry can be generalized by its resultant contact triangle relationship, $\mathcal{T}$ . (b) The response of a tendon-driven underactuated finger given actuation is dependent on spring constants and pulley radii. . . . .	103
5.6	(a) Peg insertion is viewed as a planar challenge. (b) For insertion via purely rotational motion about $\mathcal{X}$ , the peg’s edge is aligned directly above the hole with a starting angle, $\beta_0$ . (c) Translating $\mathcal{M}$ downward to $\delta$ guarantees that a pure rotation will align the peg with the hole, where $\delta_c$ encourages premature contact and leverages compliance to aid in alignment. (d) Upon rotation about $\mathcal{X}$ , the peg must overcome contact constraints (red friction cones) of the hole contacts to align for insertion, while aided by virtual spring forces $k_c$ supplied by compliance. . . . .	104
5.7	Experimental objects considered in the <b>Tight Tolerance Tasks</b> and <b>Open World Tasks</b> . All lengths are in <i>mm</i> and points on object faces indicate PCA-determined edge manipulation frames. (a) small circle, (b) large circle, (c) pear, (d) triangle, (e) rectangle, (f) YCB <i>004_sugar_box</i> , (g) YCB <i>008_pudding_box</i> , (h) YCB <i>009_gelatin_box</i> , (i) YCB <i>040_large_marker</i> , (j) green charger. (k)-(n) YCB <i>065_cups</i> . . . . .	109
5.8	(a) System setup overview. (b) Tight tolerance insertion of 5 peg geometries, highlighting the observational/external view and the tracked 6D pose via the RGBD camera. (c) System ablations include reducing compliance of the hand and the arm, in addition to deliberately adding noise to the pose estimate. (d) The open world task of plug insertion is highlighted, showcasing the sequence of actions taken from grasp to insertion. (e) Five other open world tasks were also evaluated – box packing, marker insertion, and cup stacking. Please refer to the supplementary video for a complete overview of evaluated tasks. . . . .	114

5.9	Object insertion can be conceptualized as the continual addition and modulation through time of an object’s constrained degrees-of-freedom. By continually modulating forces once constraints are detected, tight tolerance insertion can be achieved without <i>a priori</i> knowledge of the object geometry or exact hole pose for convex objects. . . . .	118
5.10	The Contact Formation-space (CF-space) of an insertion task can be conceptualized as an explicit organization of different contact formations that share borders according to constraint similarity and possible transitions. Transitions between contact formations are represented when constraints are added or removed to the state of an object. By controlling paths through a CF-space from light (few constraints) to dark (more constraints), the object follows a progression towards insertion. <b>Note:</b> This depiction of contact types is not exhaustive and other intermediate formations may be possible. . . . .	122
5.11	A Yale OpenHand Model O and a 6-axis force/torque sensor are affixed to the end of Barrett WAM manipulator. Inside of the palm of the hand, an in-hand camera setup is fabricated as to monitor the state of the object during manipulation via an AprilTag. . . . .	127
5.12	Tested objects can be classified into three categories: Tight Tolerance Tasks, Insertion Toy, and NIST Assembly Task Board. Objects are referenced according to their face geometries: (a) circle, (b) pear, (c) large triangle, (d) rectangle, (e) cube, (f) small triangle, and (g) clove. Subfigure (h) illustrates the insertion toy’s hole layout with designated search spaces in green. NIST objects (i) and (j) are referred to as the plug and gear, respectively. . . . .	128
5.13	An object is pushed along the (a) <i>x</i> –, (b) <i>y</i> –, and (c) <i>z</i> – <i>axes</i> with a linear pusher to evaluate the <i>force plateau</i> along each dimension, or more specifically, the amount of force the hand can resist before slip occurs. . . . .	128



5.14	The progression of insertion is depicted in (a), where an object goes from free space (0 constraints) to inserted (5 constraints) into its goal configuration. (b) During this process, forces are modulated and added through different steps in the insertion task (forces are smoothed and placed in the world frame for clarity). (c) We evaluate this algorithm with tight tolerance tasks, insertion toys, and objects from the NIST Task Board. . . . .	131
A.1	The proposed system is robust to external disturbances imposed during task execution. In this task sequence, the pose of the hole, arm, and object are all manually perturbed and the system recovers such that the object successfully reaches its goal configuration. Refer to the supplementary video for the complete results. . . . .	138
A.2	The ablations make use of two manipulators with different levels of compliance at the arm, hand, or environment level. (a) A Barrett WAM serves as a low-impedance, compliant manipulator and has been tested with a compliant (left) and a rigid (right) hand; while the (b) Kuka IIWA is an example of a rigid manipulator, which has been tested with a rigid hand and a compliant (left) or rigid (right) hole. . . . .	139

# List of Tables

2.1	CHAPTER 2 NOMENCLATURE . . . . .	10
2.2	SIMULATED HAND PARAMETERS . . . . .	23
2.3	FIVE-FOLD CROSS VALIDATION SCORES ON TRAINING SET . . . . .	38
2.4	FEATURE SETS DETERMINED BY FEATURE REDUCTION . . . . .	42
2.5	CLASSIFICATION ACCURACY WITH DIFFERING FEATURE SETS . . . . .	42
3.1	CHAPTER 3 NOMENCLATURE . . . . .	48
3.2	OBJECT PROPERTIES FOR THOSE USED IN SIMULATION . . . . .	62
3.3	EXPERIMENT PARAMETERS . . . . .	66
4.1	CHAPTER 4 NOMENCLATURE . . . . .	71
4.2	EXPERIMENT EVALUATION . . . . .	90
5.1	TIGHT TOLERANCE INSERTION RESULTS . . . . .	110
5.2	SYSTEM ABLATION ANALYSIS . . . . .	112
5.3	METRICS FOR OBJECT INSERTION EXPERIMENTS . . . . .	130

# Acknowledgements

Well, this acknowledgements section is more for me than it is for you. So buckle up friends, because this has been a long and wild ride and I have just a few people I would like to thank.

Well, first and foremost, I would first like to thank my advisor, Aaron Dollar, for his years of mentorship on and off the court. The one who took a chance with me all the way back in January of 2017 when I was graduating from that “small state school”, Youngstown State University. I still remember opening the offer letter to this day. Yet, I must admit, I felt like an imposter for the first several months starting this journey, but little by little, I persevered. He gave me freedom during my PhD to ask questions—usually the wrong ones—and seek out solutions—though usually the wrong solutions. I am thankful for his willingness to deal with me throughout these past six years, and more importantly, his perspectives on practical solutions to robot manipulation. I would also like to extend a huge thank you to my other committee members—Brian Scassellati and Marynel Vázquez—for their kind remarks and mentorship along the way. Scaz, your advice was always kind and insightful, and Marynel, your enthusiasm was encouraging in every conversation we had and I always left a meeting with a mind full of new ideas.

I would certainly be remiss, before I get into the longer list of acknowledgements, to not thank the two that I would say made all of this technical work possible, Kaiyu Hang and Walter Bircher. Around the time of my Area Exam in November 2019, the three of us banded together with ideas, passion, and jubilation to create the group that Hang referred to as Quantum Manipulation. While I still wholeheartedly disagree with the roots of the name Hang made up for this group, the three of us (willfully! and! efficiently!) collaborated

together for three years and produced some pretty encouraging research. Hang, you have made such an impact in my life and my research career. You have encouraged me to ask new questions, and well, question my own actual thinking. We have organized academic workshops together, conceptualized research grants together, and even had a bit of fun along the way. You are an incredible researcher, mentor, and friend—and I am so thankful to have spent several years working alongside you. Walley, from day one after my Area Exam, you and I were thinking of new projects and ways to get half-ideated methods to work. I can't believe what we have all been able to accomplish in the (admittedly) short time we had together. I can't thank you two enough for both your friendship, and your willingness to discuss mechanics with me at any time of the day or night.

Wow, how is this already over a page? Huh, well still going along the Yale trajectory, my labmates have been instrumental in defining my time here. Sam with his hot pots (and cute cats). Hector with his MIT curiosity and criticism. Vatsal with his, well, very thoughtful and insightful methods, solutions, and acknowledgements for anything in life or engineering. Mike Leddy for his jubilancy, friendship, and ATI force-torque sensor code. Yuri for his questioning of anything known to man. Berk Calli for being my first in-lab mentor, and a dear friend. Ad Spiers (Señor Peludo) for his sick beats getting me through paper writing. And finally, the great Neil Bajaj, the friend who knew just a bit about everything, but most importantly, how to beat the the last mission in Halo 2. I admire you all.

Beyond those at Yale, I would not have even started a PhD without acknowledging the educators who inspired me to be where I am today. From Youngstown State University, I would like to acknowledge Amy Cossentino, Ronald Shaklee, and Kerry Meyers. Their love for learning and teaching was always something that motivated me to be a better student and educator, and I have been so thankful for their mentorship and friendship along the way. But even before that, educators and coaches at Mathews High School such as Lou DeMarco, George Garrett, Jeff Parent, Michael Snyder, Jason Lee, the late Dan Kennedy, and Jared Terlecky have had a profound impact on my career trajectory beyond words. My upbringing has taught me many things and has brought me to this important point in my life. I am so thankful for all of your help and insight along the way.

Throughout this journey, my friends have been one of my major support systems. Ever

since our time at YSU, my friendship with Pabst, Briggs, and Maddi has only gotten stronger. I can't thank them enough for their frequent weekend visits, their words of encouragement, and their continued support for me along the way. I can't underscore this enough. Additionally, I want to thank my undergraduate roommates Brock, Hova, Nico, and Ryan who have continued to support me throughout this prolonged academic journey. Who would have thought we would all be where we are today? And finally, I want to thank the friends I did acquire during my time at Yale. Lara, you have always been supporting since we met and convincing me I actually know something about engineering. Those little tidbits always kept me going. And, while it would be impossible to list the many engineers, Murray affiliates, and intramural teammates I interacted with, I would especially like to thank my dear friends Ariye, Cody, Dan, David, Hailey, Jack, Jade, Josh, Mason, Michelle, Nadeen, and Stephen for their energy, their kindness, and their memes along the way. I am so thankful I was able to spend time with you all during this journey, and I look forward to our continued crazy adventures in the future.

Last and surely not least, I would like to thank my family—Mom, Dad, Chris, Kayla, Luke, and Elle—for their support and love. I wouldn't have traded the numerous trips back to Ohio for weddings, birthdays, holidays, and baptisms for the world. Your love and support for me along the way, in addition to all from my aunts, uncles, and cousins, was something that always kept me going and served as a motivation to complete this thesis. They often say you do your PhD for yourself. Well, I did this PhD for me, but I also did it for my whole family – especially my grandparents, who I am sure are up there today still trying to read and understand this thesis. Thank you all for your love and support.

# Chapter 1

## Introduction

*~ Complex robots complicate control; keep designs simple and exploit emergent behavior ~*

### 1.1 Motivation for Society

The dexterous capabilities afforded to humans by our hands are unparalleled to that of other species, allowing us to complete an array of daily manipulation tasks with ease [1]. As an illustration, let's examine the task of inserting a key into a lock. First, the hand must successfully acquire a stable grasp on the key, properly selecting contacts and maintaining them online via proprioceptive and tactile feedback. Once the contacts are stable and force modulation is achieved, the key must then be reconfigured within-hand via coordinated finger motions – balancing forces while making and breaking contact during a finger gaiting process. Upon reorientation of the key, the hand-object system then evaluates how forces must be applied by the key into the lock tumbler for successful insertion and then finally rotation.

This task decomposition serves as a single elaboration of the complex manipulation tasks humans mindlessly complete daily; seamlessly combining sensing, control, and planning. Numerous other tasks can be similarly decomposed, e.g., washing dishes [2], inserting batteries into electronics [3], or preparing food in the kitchen [4]. In emphasizing this narrative, service robots of the future must be able to complete a similar array of “everyday”

tasks as that of a human, which has been unrealized by robots to date. In this thesis, I will elaborate on what we feel is a promising approach towards establishing these capabilities for robot hands and the role I believe compliance will play in this process.

## 1.2 The Human Hand

A great deal of the human hand’s manipulation capability relies on tactile feedback from touch sensors located all over our hands. In fact, works dating back decades ago [5], investigated just how well our hands are able to manipulate objects when this sensation is muted, even for a brief period of time. While this complicated tasks, it did not make all manipulation impossible. One observation was just how important vision becomes when touch is muted—and how we as humans innately have some sort of “fusion” between our sensing modalities.

While still not completely understood, it is believed that approximately 40% of the motor and sensory cortex of our brains in humans is devoted just to our hands. And from that, it is easy to take for granted just how complex these body parts are and how vital of a role they have played to human evolution. As eloquently said in [1]:

“

In one of his books on nature sciences, the greek philosopher Aristotle (384–322 BC) thus argued against the conceptions of his late colleague Anaxagoras (500?–428 BC ) regarding the relationship between human hands and mind. As they appear to be the two most distinguished features of humans among animals, the two philosophers debated whether it was because humans had dexterous hands that they became intelligent, or the other way around. Anaxagoras’ intuition has been later on confirmed by several findings of paleoanthropologists, showing that the mechanical dexterity of the human hand has been a major factor in allowing homo sapiens to develop a superior brain.

”

Yet one observation this quote does not delve into, is just how much of a vital role the human hand’s compliance likely played over centuries. In particular, as a fun exercise, note the amount of flexure your distal phalanges are able to reconfigure when a force is applied to your fingertip—how far can it be pulled backwards? Structurally, this reconfiguration given an external force or perturbation serves as a mechanism for protecting your fingertips from damage when grasping in unknown environments, without sensory information, or maybe just carelessly. The “buckling” nature of this phalange has been dynamically studied [6] and it is likely that without this mechanism, the human hand would not look the same way it does today.

### 1.3 Research Overview

So if humans have some amount of variability in their hand kinematics, and are able to control a fair number of objects without sensing touch, why can’t robots? This general realization serves as a basis for the work outlined hereafter. Particularly, I am interested in understanding the extent to which we can control robot hands that are both, mechanically compliant and sensorless, i.e., lacking of tactile sensors and joint encoders. How can we observe the state of the hand-object system for these types of robots? How can we understand and control “parasitic” motions associated with these mechanisms? How much does the compliance of these mechanisms help the planning process? And finally, is vision enough for complex in-hand manipulation and whole-arm assembly tasks?

To keep this research tractable, my work will mostly focus on fingertip-based in-hand manipulation. I choose to delve into this specific class of manipulation because it will allow us to formulate complete grasp mechanics-based representations that would theoretically represent any hand-like end effector, or in other words, a single mechanism with more than one serial link manipulator, i.e., fingers. Moreover, these studies concerning fingertip-based in-hand manipulation can serve as building blocks for more advanced whole-hand manipulation capabilities that do not rely solely on fingertip-only contact, e.g., [7].



## 1.4 Thesis Overview

The narrative progression of this thesis will follow the story arc associated with this aforementioned research overview. Specifically, we will begin by discussing grasp mechanics and introducing some representations that can particularly encapsulate the kinematics associated with rigid mechanisms and contact mechanics through vision. Thereafter, I will discuss how these representations are beneficial for observing the state of a hand-object system. In particular, this formulation allows us to generalize learning for observing the state of underactuated hands to a certain extent, which is beneficial for transferring learned models between grippers. This will be discussed in Chapter 2.

Once we can observe the system, we can then focus our work in controlling such robots. In particular, I will focus on being able to control fixed-contact manipulation with a three-fingered underactuated gripper. Given desired trajectories, that could be fully- or partially-constrained, we develop a control algorithm based on Model Predictive Control that is able to choose actions for the robot. This method benefits from others in the literature in that we can alter constraints between different goal trajectories and also run our algorithm online via visual feedback, ultimately allowing us to recover from unmodeled external perturbations. This will be discussed in Chapter 3.

Planning complex motions as to extend the workspace of the gripper becomes a large focus thereafter. Specifically, we investigate the use of finger gaiting with an underactuated hand and develop a multi-modal planning algorithm that leverages the lessons learned from fixed-contact manipulation. Similar to before, our method is a fast, online algorithm that relies solely on vision, and can thus recover from unmodeled external perturbations. Our contributions from this study introduce the overarching theme for this thesis – compliant robots allow us to accelerate both learning and planning. We show this finding from a robot planning perspective for the multi-modal case, and discuss how modal transfer configurations turn into modal transfer regions due to the passive adaptive ability of the robot. This will be discussed in Chapter 4.

Finally, to end this thesis, in Chapter 5 I investigate how our methods and models can be leveraged for complex, whole-arm assembly tasks. Particularly, we study how vision

and compliance can work together to complete traditionally difficult or even historically impossible insertion tasks for robots. These experiments I complete underscore just how valuable of a role we believe compliance will play in the future of robot manipulation.

## Chapter 2

# Observing the State of a Complaint Hand

### 2.1 Introduction

Developing robots capable of performing tasks in human-made, unstructured environments has remained an overarching research question in robotics for several decades. An important building block to this question is addressing the development of dexterous, within-hand manipulation (WIHM) capabilities for robotic hands. Dexterous manipulation is often characterized as the skillful, coordinated use of an end effector to reposition or reorient an object with respect to the hand frame [1]. An example of this ability includes the task of removing a key from a pocket, reorienting, and inserting into a lock. In this task, not only does WIHM enable repositioning and reorientation of the key without re-grasping or large whole-arm motions, it also allows the robot to avoid undesired system conditions, such as diverting away from joint singularities, while attempting to repose the key [8]. WIHM capabilities are especially advantageous for more capable service/home robots—which would be required to perform a variety of daily activities, such as folding clothes or feeding humans [4, 9].

Practical implementation of precision WIHM remains a major challenge as coordinated finger movements with rigid, high degree-of-freedom hands requires accurate hand-object

models, accurate parameter estimations of the environment, and advanced control schemas, which may be impossible to derive or estimate. An alternative to WIHM with fully actuated hands has been through the use of soft, compliant, or underactuated grippers that are able to passively adapt to their environment. While this ability enables grippers to more easily handle sensing and perception uncertainty [10], it also introduces difficulties in modeling—the configuration of the hand is dependent on fingertip forces, joint stiffnesses, and contact locations, which may be impossible to accurately measure.

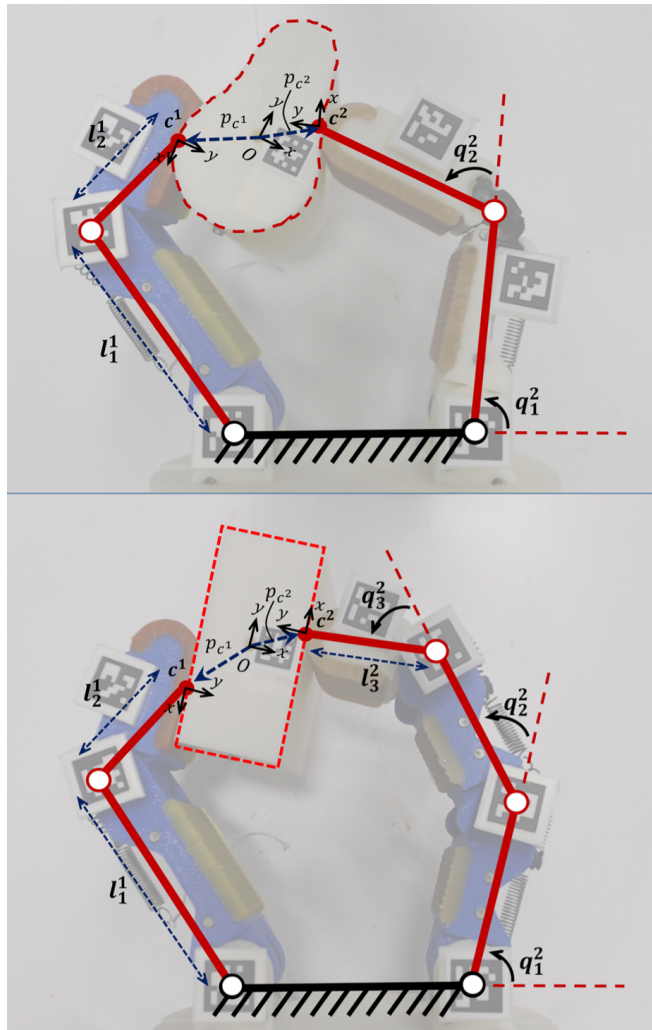


Figure 2.1: Geometric features can be extracted visually during manipulation with a priori knowledge of the fingertip geometry, object geometry, and the number of finger links. (Top) A pivot-flexure finger manipulates a pear-shaped object with rolling contacts (Mode: Normal). (Bottom) A three-link pivot finger manipulates a rectangular object until sliding occurs along the left finger (Mode: Sliding).

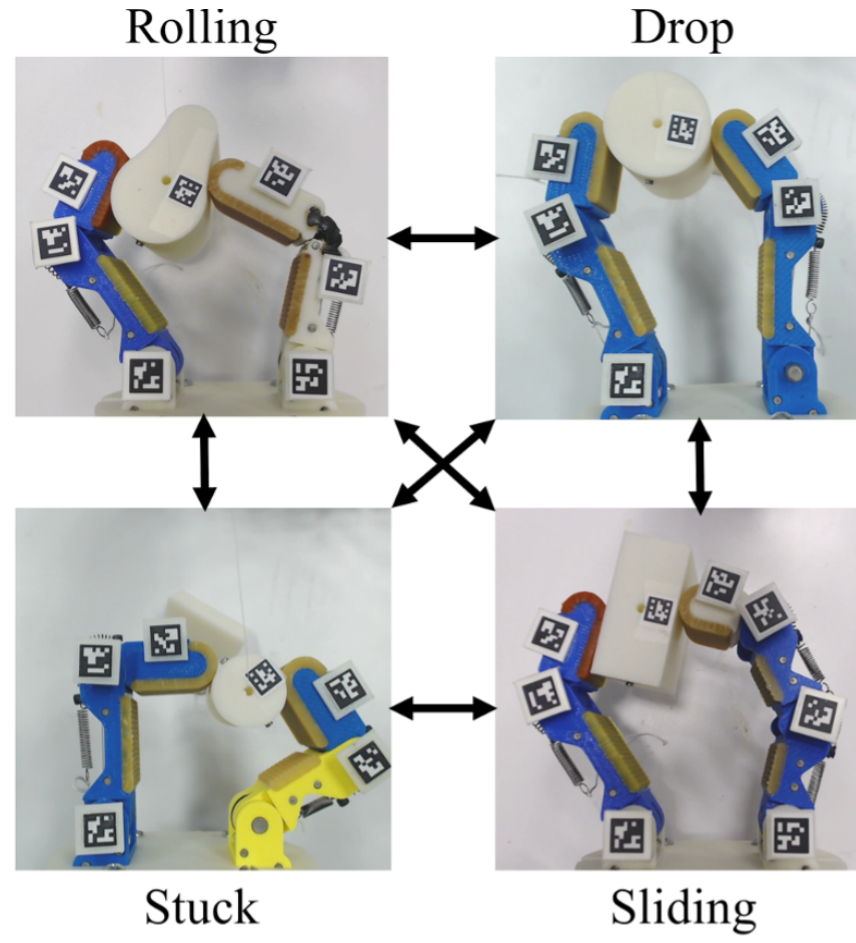


Figure 2.2: By predicting modes of manipulation before or at the moment they occur, the user is able to transition between modes (trigger or avoid) for desired manipulation. Modes are typically detected when the hand-object system is in a similar configuration as those shown.

Due to analytical modeling difficulties, machine learning has been introduced into manipulation for both, fully-actuated and underactuated hands. Such approaches are able to intrinsically estimate parameters, e.g., kinetic model parameters or joint stiffness ratios, that can be difficult or impossible to model via human intervention. While these approaches can be fairly successful, they often rely on large, unstructured feature sets for training, e.g., camera or tactile array inputs [11]. In this approach, little intuition is provided by the learned model as to what characteristics of the system are most valuable for the task.

In this chapter, we address this drawback by utilizing grounded, mechanics-based features that are able to generalize to different system variants. Assuming quasistatic mo-

tion of both the hand and the object, purely geometric representations—including finger manipulability measures, grasp quality measures, and hand-object manipulability measures—constitute as elemental, generalized properties of the hand-object system (Fig. 2.1). We investigate how these features allow trained models to transfer more successfully than traditional joint-based features. We use these features to distinguish between four possible manipulation classes for fingertip-based, prehensile manipulation; namely, normal (rolling contact), drop, stuck, and sliding. These classes, coined as modes of manipulation for this work, can be predicted through a self-supervised learning approach—which would enable the user to either trigger or avoid modes for desired object movement (Fig. 2.2), as in [12, 13].

The approach of using mechanics-based features is particularly advantageous for generalizing models among a task [14]. Due to reliance on the underlying mechanics of the problem, a single classifier can be self-supervised and trained on one gripper variant and then transferred to another similar but distinct variant without retraining or data adaptation. We theoretically explore this concept and show the bounds by which hand parameters can change before mode distributions of the features become distinct between variants. We also test this experimentally by using estimated Cartesian motion models to randomly manipulate different objects, and self-tagging each of the modes when they occur. A classifier is trained offline using a single gripper variant, and we show the transferability of the learned model for 5 different, asymmetric hands. In the continuation of this chapter, please refer to the Table 2.1 to utilized nomenclature:

Table 2.1: CHAPTER 2 NOMENCLATURE

Symbol	Description
<b>General:</b>	
$q$	Particular hand configuration: $q \in \mathbb{R}^4$ or $\mathbb{R}^5$
$a$	Configuration of the actuators: $a \in \mathbb{R}^2$
$B, F, O$	Pose of the base frame, finger frame, or object frame, respectively: $B, F, O \in SE(2)$
$v$	Velocity = $(v_x, v_y, v_\theta) \in se(2)$ of the object w.r.t. $B$
$J^i$	Jacobian of the $i^{th}$ finger of the hand: left finger is index 1 and right finger is index 2. $J_h$ is the Hand Jacobian.
$G$	Grasp Matrix: $G \in \mathbb{R}^{3 \times 4}$ in the two-finger, planar case
$H$	Hand-Object Jacobian: $H \in \mathbb{R}^{3 \times 4}$ or $\mathbb{R}^{3 \times 5}$
$\mathcal{P}$	Object point cloud ( $\mathcal{P}_o \in \mathbb{R}^{2 \times N}$ ) w.r.t. $O$ or fingerpad point cloud ( $\mathcal{P}_f \in \mathbb{R}^{2 \times N}$ ) w.r.t. $F$
<b>Grasp Mechanics-based Features:</b>	
$v$	Cartesian velocity reference of the object: $v_x$ in the $x$ -direction and $v_y$ in the $y$ -direction.
$w^i$	Manipulability measure of the $i^{th}$ finger: $w_p^i$ is the penalized manipulability measure
$g$	Singular values (SV) of $G$ : $g_{max}$ max SV, $g_{min}$ min SV
$h$	Singular values of $H$ : $h_{max}$ max SV, $h_{min}$ min SV
$c^i$	Curvature of the contact point on the $i^{th}$ finger: $c_f^i$ is fingerpad curvature and $c_o^i$ is object curvature

## 2.2 Related Work

In this section, we present traditional methods to modeling within-hand manipulation. Following, we cover recent approaches to learning manipulation and this method’s associated drawbacks, which motivates this work.

### Within-hand Manipulation

For several decades, a great deal of research in robot manipulation has focused on explicitly modeling physical interactions that occur between robots and objects in complex, unstructured environments—from fundamentals of interactions such as pushing [15], to object interactions in highly dynamic and unconstrained environments. The study of these interactions is especially entailed in the application of WIHM, that requires coordinated

finger movements while maintaining predefined contact scenarios. Since Okada first used inverse kinematics to plan joint trajectories for manipulator motion almost four decades ago [16], nearly every aspect of robot manipulation has been treated with great mathematical rigor in the pursuit of creating more capable robots [17–22]. This great volume of work elucidates many powerful relationships between finger joint motion and object motion via classic formulations such as contact curvatures, the Grasp Matrix, the Hand Jacobian, and the Hand-Object Jacobian.

Leveraging these mechanical representations and assuming that specific contact models are warranted by the task, object motion models can be devised. The point contact with friction model, denoting that forces can be exerted in any direction within the friction cone, is often used and has led to the formulation of the Hand-Object Jacobian [23], which represents the transition map from joint movement into object motion. The work in [22] assumed stationary point contacts without rolling or sliding, virtually fixing the location of the contact frame on the finger to that on the object. Rolling has been taken into account as well [24–26], which requires geometric knowledge of the fingerpad and the contact to maintain precision. More advanced contact models, such as those for soft contacts [27], have also been introduced. Generally, all contact models are highly subject to material parameters, such as durometer and texture of the contact, that can change with environmental conditions (humidity or dust). Therefore, parametric estimation often necessitates on-board sensors, which are expensive, inaccurate, and complicate the design and control of the hand.

The addition of compliance to the system through either hardware (soft, underactuated) or software (impedance control [28], soft synergies [29]) can help mitigate uncertainties that would otherwise lead to task failure. Both approaches introduce passive adaptability to the system, which permits a grasp to be maintained under reasonable external disturbances [30]. For example, in [31], this concept is leveraged for in-hand manipulation with simple control over just two degrees of actuation. Though, due to this compliance, precision manipulation remains difficult to accurately model or simulate, since the output space is of higher dimension than the input space [26, 32, 33].

Two promising approaches to planar precision manipulation with underactuated hands



have been introduced in our previous works by either using rough gripper models and an MPC visual servoing framework [34], or learning a state transition model of the gripper [35]. Although object precision was increased in both works, manipulation was focused in a specific region of the workspace. Moreover, the models learned were system specific and transfer was not addressed in either of these works providing inspiration for this manuscript—to learn transferrable representations of the gripper to aid in generalizing manipulation.

### **Learning Manipulation Policies**

Learning control policies for dexterous manipulation is a well-studied research area when analytical representations are unavailable. This approach enables the robot to formulate its own representative model without hand-tuned, human intervention. Reinforcement Learning (RL) has shown to be a promising approach to this problem, especially for compliant systems, e.g., [36]. A major drawback to RL is the amount of data required to train the model. As presented in [11], over “a hundred years” of object manipulation was collected in simulation for WIHM of a cube. Though some approaches have addressed this caveat, e.g., by learning from online videos [37] or guiding the manipulation strategies by combining imitation learning of a human expert [38,39], simulators, which are often not representative of real-world contact scenarios, are normally required to develop these learned models. In addition to these drawbacks, the input dimensionality used in multilayer perceptrons can be extremely large and will therefore lack interpretability and generalizability for a more enlightened approach to manipulation. For example, in [11], the input vector was a video stream from 3 cameras (thus,  $3 \times 640 \times 480 = 921,600$  pixels/features).

Aside from learning the entire system model for precision manipulation, detecting object phenomena such as sliding has also been reported in the literature. Previous works have learned from tactile “images” to detect the coefficient of friction at the point of incipient slippage [40,41] and can therefore plan trajectories to avoid slip conditions [42,43]. Unfortunately, due to the nature of this approach, prior exploration with the object is necessary, which may be infeasible for time-sensitive or mission critical tasks. By leveraging mechanics, slip conditions can also be avoided by ensuring reasonable grasp quality measure values during manipulation [44]. Nevertheless, compliant, soft, and underactuated hands are often

not equipped with the sensing modalities required to detect such phenomena, providing inspiration and purpose for this work.

## 2.3 Grasp Mechanics-Based Features

In this work, we leverage traditional mechanical models of manipulation to define features generalizable to different hand variants. Specifically, we extract the most common manipulability measures associated with the hand, the object, and the contacts: a Jacobian-based manipulation measure, a penalized Jacobian-based manipulability measure, the singular values of the Grasp Matrix, the singular values of the Hand-Object Jacobian, and the contact curvatures. By learning from these features, which are grounded geometrically to the state of the gripper, we are able to analyze which traditional grasp-mechanics measures are able to best represent the hand-object state. A summary of this manuscript’s nomenclature is presented in Table 2.1. For the remainder of this manuscript, we will refer to the left finger as index 1 and the right as index 2.

For the following formulations, let’s assume the planar manipulator has  $n$  serial-link fingers, each having  $j^i$  joints per finger. The configuration of a single finger,  $q^i \in \mathbb{R}^{j^i}$ , represents its current joint angles. Therefore, the hand configuration,  $q \in \mathbb{R}^{\sum_{i=1}^n j^i}$ , fully describes the state of the hand and denotes the angles associated with each of the joints (Fig. 2.1). In traditional modeling for grasping and manipulation, the Hand Jacobian, sometimes referred to as the Manipulator Jacobian, is denoted as  $J_h = \text{blkdiag}(J^1, \dots, J^k)$ , where  $J^i$  is the Jacobian for a single, serial-link finger. It is important to note that in underactuated or compliant systems, the configuration of the gripper cannot be fully described by the state of the actuators,  $a$ . That is, it is likely the  $\dim(a) < \dim(q)$ .

### 2.3.1 Finger Manipulability Measures

The current configuration of a serial-link finger determines its manipulability, i.e., how the tip of the finger is able to move given an actuation input about each of the joints, and is represented by the finger Jacobian. This representation is a function from joint input velocity to fingertip velocity:

$$y^i = J^i \dot{q}^i \quad (2.1)$$

where  $y^i \in \mathbb{R}^2$  in the planar case ( $y^i \in \mathbb{R}^3$  in the spatial case),  $J^i \in \mathbb{R}^{(2 \times j \times i)}$  in the planar case ( $J^i \in \mathbb{R}^{(3 \times j \times i)}$  in the spatial case), and  $q^i \in \mathbb{R}^{j^i}$ . In this work, we utilize both, two-link and three-link serial manipulators in the plane. From this, we can formulate the two finger Jacobians:

Two-link finger:

$$J^i(q^i) = \begin{bmatrix} -\mathcal{J}_A - \mathcal{J}_B & -\mathcal{J}_B \\ \mathcal{J}_D + \mathcal{J}_E & \mathcal{J}_E \end{bmatrix} \quad (2.2)$$

Three-link finger:

$$J^i(q^i) = \begin{bmatrix} -\mathcal{J}_A - \mathcal{J}_B - \mathcal{J}_C & -\mathcal{J}_B - \mathcal{J}_C & -\mathcal{J}_C \\ \mathcal{J}_D + \mathcal{J}_E + \mathcal{J}_F & \mathcal{J}_E + \mathcal{J}_F & \mathcal{J}_F \end{bmatrix} \quad (2.3)$$

$$\mathcal{J}_A = l_1^i \sin(q_1^i)$$

$$\mathcal{J}_B = l_2^i \sin(q_1^i + q_2^i)$$

$$\mathcal{J}_C = l_3^i \sin(q_1^i + q_2^i + q_3^i)$$

$$\mathcal{J}_D = l_1^i \cos(q_1^i)$$

$$\mathcal{J}_E = l_2^i \cos(q_1^i + q_2^i)$$

$$\mathcal{J}_F = l_3^i \cos(q_1^i + q_2^i + q_3^i)$$

where, more specifically  $q^i = [q_1^i, \dots, q_j^i]^T$ , which represents the joint configuration for a single finger,  $i$ . From this Jacobian, we can represent its manipulability measure,  $w^i$ , for each finger in the hand [45]:

$$w^i = \sqrt{\det(J^i * \text{transpose}(J^i))} \quad (2.4)$$

As  $w^i$  approaches zero, this is indicative of the individual mechanism nearing a singularity—which effectively limits the ability to instantaneously move in any direction.

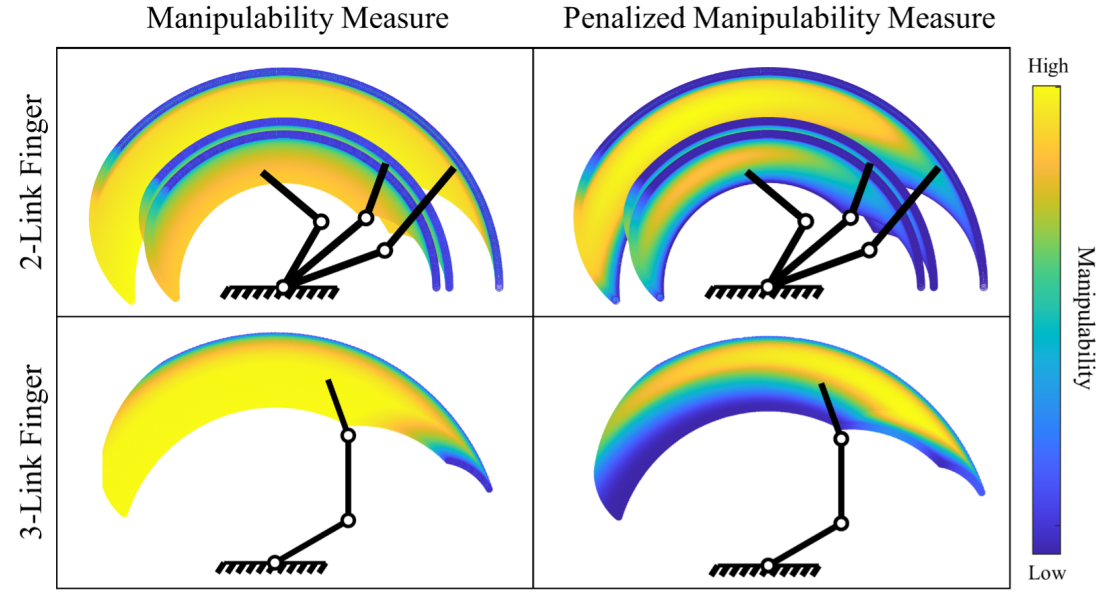


Figure 2.3: Free swing manipulability workspaces for both proposed manipulability measures. (Top) Free swing trajectories for three two-link planar fingers used in this work. (Bottom) Free swing trajectory of the three-link planar finger used in this work. Unlike the penalized manipulability measure, the standard manipulability measure does not account for joint hard stops.

A penalized manipulability measure is also proposed, as it better encapsulates limits of a finger’s workspace by incorporating a priori knowledge of the hard stops, i.e., a finger link cannot rotate fully around a joint, but typically has a range in which it can operate [46]. Fundamentally, this measure enables the mechanism to determine where mechanical constraints are located and to stay well within the workspace. The penalized manipulability measure,  $w_p^i$ , is the product of a penalty value,  $\pi^i(q^i)$  and the manipulability measure from (2.4).

$$\pi^i(q^i) = 1 - e^{-\kappa \prod_j \frac{(q_j^i - l_j^{i-})(l_j^{i+} - q_j^i)}{(l_j^{i+} - l_j^{i-})^2}} \quad (2.5)$$

Here,  $l_j^{i+}$  and  $l_j^{i-}$  represent the upper and lower bounds on joint  $j$ , respectively, and  $\kappa$  is a weighting factor that is tuned to determine how quickly manipulability drops off near the joint limits. This penalty function is calculated for each finger, and is applied to determine  $w_p^i$ ,

$$w_p^i = \pi^i w^i \quad (2.6)$$

The two finger manipulability measures,  $w^i$  and  $w_p^i$ , are used as mechanics-based features for mode detection in this work. We provide an illustration of these measures in Fig. 2.3, where it is important to note the similarities of the data distributions as properties of the fingers change.

### 2.3.2 Grasp Quality Measures

A grasp quality measure based on the Grasp Matrix is utilized for representing the manipulability of the object, given the current contact configuration. The Grasp Matrix is commonly leveraged as a representation for relating the velocity of the contact to the velocity of the object. Determined by the contact normal directions, in addition to the relative location of the object’s fixed frame, the Grasp Matrix is formulated strictly by the geometry of the object and the position of the contacts—force sensing is not required. The model has a desirable quality that, even though the upper bound of its singular values is unbounded, the minimum singular value has a lower bound of zero regardless of the dimensions of the object. This occurs when two or more contact normals are collinear, parallel vectors with respect to the object frame,  $O$ . This quality can be a useful indicator for when the object is likely to drop. A metric based off of singular values of the Grasp Matrix is therefore invariant across systems of different dimensions.

The Grasp Matrix,  $G$ , in the velocity domain represents a map from external contact velocities,  $\dot{z}$ , to object frame velocity,  $v$ . We can represent this as:

$$\dot{z} = G^T v \quad (2.7)$$

The shape of  $G$  is not absolute as it depends on the contact model used for manipulation. In the planar case with a point contact model and  $c$  number of contacts,  $\dot{z} \in \mathbb{R}^{2c}$ ,  $v \in se(2)$ , thus  $G \in \mathbb{R}^{3 \times 2c}$ . Similarly, for the spatial case not covered in this work,  $\dot{z} \in \mathbb{R}^{3c}$ ,  $v \in se(3)$ , and  $G \in \mathbb{R}^{6 \times 3c}$ . In this work, we will assume a point contact with friction model, forming the basis,  $b_{c^i}$ , which states that a force can be applied along the  $x$ - and  $y$ -axes of the

contact accordingly so long as it is within the friction cone. Additionally, we must calculate the vector,  $p_{c^i}$ , denoting the positional relationship between the contact frame,  $c^i$ , for the  $i^{th}$  finger and the object frame,  $O$ . The rotational relationship,  $\theta_{\delta^i}$ , between the contact frame,  $c^i$ , and  $O$  is also computed. For the two-finger, two-contact case in this work:

$$b_{c^i} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} \quad (2.8)$$

$$R_{c^i} = \begin{bmatrix} \cos(\theta_{\delta^i}) & -\sin(\theta_{\delta^i}) \\ \sin(\theta_{\delta^i}) & \cos(\theta_{\delta^i}) \end{bmatrix} \quad (2.9)$$

$$p_{c^i} = \begin{bmatrix} p_{c^i x} \\ p_{c^i y} \end{bmatrix} \quad (2.10)$$

where  $\theta_{\delta^i} = \theta_{c^i} - \theta_O$ , and  $\theta_{c^i}, \theta_O$  are the angle offsets of the  $i^{th}$  contact frame and the object frame, respectively. Finally, with these calculated for each contact, we can formulate the Grasp Matrix,  $G$ :

$$Ad_{g_{oc^i}}^T = \begin{bmatrix} R_{c^i} & 0 \\ \begin{bmatrix} -p_{c^i y} & p_{c^i x} \end{bmatrix} R_{c^i} & 1 \end{bmatrix} \in \mathbb{R}^{3 \times 3} \quad (2.11)$$

$$G = \begin{bmatrix} Ad_{g_{oc^1}}^T b_{c^1} & Ad_{g_{oc^2}}^T b_{c^2} \end{bmatrix} \in \mathbb{R}^{3 \times 4} \quad (2.12)$$

From  $G \in \mathbb{R}^{3 \times 4}$ , there exist three singular values that describe the state of the contacts with respect to the object. In our feature set, we will denote the maximum singular value as  $g_{max}$  and the minimum singular value as  $g_{min}$ .

### 2.3.3 Hand-Object Manipulability Measures

The Hand-Object Jacobian [23] is a map that describes the relationship between actuation input,  $\dot{q}$ , and object velocity,  $v$ . Although this cannot be directly utilized in underactuated hands, due to the inability to control each of the joints individually, i.e.,  $dim(a) < dim(q)$ , it's geometric representation of the hand-object system can provide insight as to where the

object can move given the current hand configuration. This Hand-Object Jacobian,  $H$ , assumes a point contact with friction model and is formulated by combining the Grasp Matrix,  $G$ , and the Hand Jacobian,  $J_h$ . Let’s examine  $\dot{y} \in \mathbb{R}^{2k}$ , or the vector of all fingertip velocities of the hand from (2.1). Let’s now also assume, that  $\dot{y} = \dot{z} \in \mathbb{R}^{2k}$ , the object contact velocities from (2.7). Assuming a point contact with friction model, this further suggests that the location of the contact does not move with respect to the object frame during manipulation, and virtually attaches the finger to the object. With this assumption, we combine (2.1) and (2.7),

$$v = (G^T)^+ J_h \dot{q} = H \dot{q} \tag{2.13}$$

where  $(G^T)^+$  is the pseudo-inverse of the transposed Grasp Matrix. In the planar case with two-links and two contacts,  $H \in \mathbb{R}^{3 \times 4}$ . Here, the singular values of  $H$  represent how close the hand-object system is to a singular configuration, i.e., the ability for the object to move instantaneously in any direction. Similar to those used for  $G$ , we will use the maximum singular value,  $\hbar_{max}$ , and the minimum singular value,  $\hbar_{min}$ , as features.

### 2.3.4 Curvature of Contact

As described by Montana [47], the geometric conditions of contact are important as they enable differentiation between contact stability and spatial stability—necessary measures to track during manipulation. To this end, we propose extracting the local conditions of the contact point for both, the fingerpad and object.

The object point cloud,  $\mathcal{P}_o$ , and fingerpad point cloud,  $\mathcal{P}_f$ , as further described in Sec. 2.5.3, are used for calculating the curvature at the contacts. Let’s require that the point clouds are contiguous; that is, neighboring indices indicate neighboring points in the cloud. Given  $\mathcal{P}_o(\mathbf{p}_o)$  and  $\mathcal{P}_f(\mathbf{p}_f)$ , determined by the KD-Tree to be the two closest points to one another in separate clouds, we calculate the curvature of the contact by evaluating their relationship to neighbors at each contact point. The curvature is therefore equal to the reciprocal of the radius of the circle that fits three neighboring points in the same point cloud. For example, let’s calculate the curvature for the object. Given neighboring points on

the object,  $\boldsymbol{p}_{-o} = \boldsymbol{p}_o - 1$  and  $\boldsymbol{p}_{+o} = \boldsymbol{p}_o + 1$ , we calculate the Euclidean distance between each of the three sets of points,  $(\mathcal{P}_o(\boldsymbol{p}_o), \mathcal{P}_o(\boldsymbol{p}_{+o}), \mathcal{P}_o(\boldsymbol{p}_{-o}))$  providing distances  $\beta_1$ ,  $\beta_2$ , and  $\beta_3$ . Then,

$$\beta_s = \frac{\beta_1 + \beta_2 + \beta_3}{2} \quad (2.14)$$

$$\lambda = \sqrt{|\beta_s(\beta_s - \beta_1)(\beta_s - \beta_2)(\beta_s - \beta_3)|} \quad (2.15)$$

$$\boldsymbol{c}_o^i = \frac{\beta_1\beta_2\beta_3}{4\lambda} \quad (2.16)$$

where  $\boldsymbol{c}_o^i$  from (2.16) is the curvature of the object at the  $i^{th}$  contact point. We can similarly calculate the curvature of fingerpad at the  $i^{th}$  contact point. These curvatures are included as mechanics-based features in this work to aid in determining object stability.

## 2.4 Bounding Feature Generalizability

The goal of this section is to investigate the bounds of which grasp mechanics-based features are able to better estimate the state of the hand-object system as physical parameters of the hand change, e.g., link lengths or spring ratios. We compare these bounds to a more traditional feature set used for learning—the joint or motor configuration of the robot. This is accomplished by modeling the mechanics of quasistatic, underactuated manipulation for a two-fingered hand. After modeling, we sequentially vary parameters of the hand-object system beyond that of its original symmetric configuration and run statistical analyses that indicate whether or not the features likely come from similar distributions between different hand variants. In order to maintain brevity and tractability of these results, this section will focus on studying solely finger manipulability measures, and will leave the additional features from Sec. 2.3 to be discussed in Sec. 2.7. Physical characteristics of the hand are referenced in Fig. 2.4.



### 2.4.1 Mechanics of Underactuated Manipulation

Underactuated hands can be modeled in terms of energy with kinematic, frictional, and actuation constraints. That is, the configuration of the hand after actuation can be determined by solving for the minimum energy configuration objective,

$$U = \frac{1}{2} \sum_i \sum_j k_j^i (q_j^i - q_{j0}^i)^2 \quad (2.17)$$

where  $q_{j0}^i$  is the rest angle of each joint,  $q_j^i$  is the current angle of each joint,  $k_j^i$  is the spring stiffness of each joint, and  $U$  is the total elastic energy of the hand. This is formulated as an optimization problem, guided by both equality and inequality constraints. The quasistatic moment about the finger's proximal joint,  $M_1^i$ , on finger  $i$  is created by both normal,  $f_N^i$ , and tangential,  $f_T^i$ , contact forces,

$$M_1^i = u_{1,eff}^i \times f_N^i + u_{1,eff}^i \times f_T^i \quad (2.18)$$

where  $u_{1,eff}^i$  is a vector from the proximal joint to the fingertip and  $\times$  is the cross product. In this formulation, we assume that the normal force vector extends along the line joining both contact points to the object. We then define the moment at the finger's distal joint,  $M_2^i$ , created by contact forces,

$$M_2^i = u_{2,eff}^i \times f_N^i + u_{2,eff}^i \times f_T^i \quad (2.19)$$

where  $u_{2,eff}^i$  is a vector from the distal joint to the fingertip.

Through this analytical modeling of  $M_1^i$  and  $M_2^i$ , we represent the moment balance at the finger's proximal joint,

$$0 = T^i r_1^i - k_1^i \Delta q_1^i + M_{1\tau}^i \quad (2.20)$$

where  $T^i$  is the force created by the tendon, wrapped about a pulley of radius  $r_1^i$ ,  $k_1^i$  is the proximal spring stiffness,  $\Delta q_1^i = q_1^i - q_{1,0}^i$  is the proximal joint angle w.r.t. its rest orientation, and  $M_{1\tau}^i$  is the out of plane component of the proximal moment vector representing the

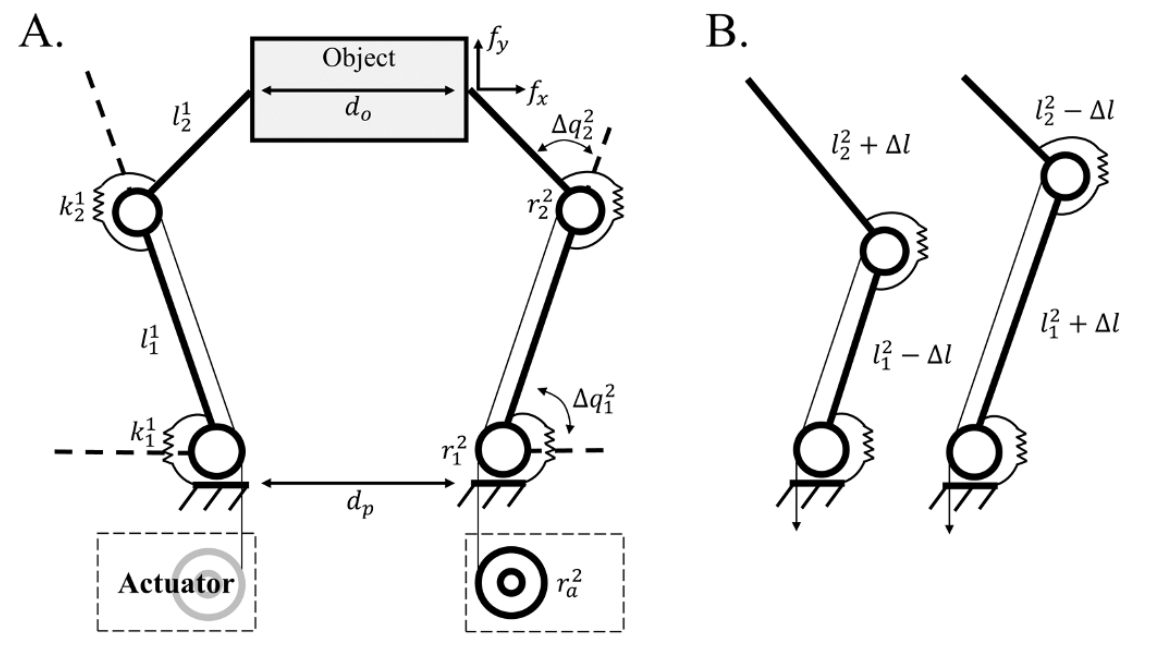


Figure 2.4: (A) Annotation of hand parameters required for modeling underactuated manipulation mechanics. (B) Proximal and distal link lengths in simulation are changed by the same value,  $\Delta l$ , as to maintain unit length during simulation.

magnitude of its torque. We similarly model the moment balance at the finger's distal joint,

$$0 = T^i r_2^i - k_2^i \Delta q_2^i + M_{2\tau}^i \quad (2.21)$$

where  $\Delta q_2^i = q_2^i - q_{2,0}^i$  is the distal joint angle w.r.t. its rest orientation.

In addition to moment balance constraints, the forces applied to the object must be in equilibrium with one another in order to maintain a stable grasp. That is,

$$0 = f_{Nx}^1 + f_{Tx}^1 + f_{Nx}^2 + f_{Tx}^2 \quad (2.22)$$

$$0 = f_{Ny}^1 + f_{Ty}^1 + f_{Ny}^2 + f_{Ty}^2 \quad (2.23)$$

where  $f_{Nx}^i$ ,  $f_{Tx}^i$ ,  $f_{Ny}^i$ , and  $f_{Ty}^i$  are the x and y components of the normal and tangential forces, respectively. During manipulation, we must also satisfy kinematic loop closure,

$$0 = \|u_{B,eff}^1 - u_{B,eff}^2\| - d_o \quad (2.24)$$

where  $u_{B,eff}^1$  is the position of the left fingertip w.r.t the base frame,  $B$ , and  $u_{B,eff}^2$  is similarly the position of the right fingertip. Here,  $d_o$  is the diameter of the object in contact with the fingertips. The last equality constraint represents the tendon or transmission constraint of underactuated mechanisms, dictating the coupled actuation between both joints,

$$0 = r_1^i \Delta q_1^i + r_2^i \Delta q_2^i - r_a^i \Delta a^i \quad (2.25)$$

where  $r_a^i$  is the radius of the actuator pulley, and  $\Delta a^i$  is the difference between the resting and set angle of the actuator. Finally, an inequality constraint on each finger must also be satisfied such that contact normal and tangential forces satisfy Coulomb's Law,

$$0 \geq \|f_T^i\| - \mu_o \|f_N^i\| \quad (2.26)$$

where  $\mu_o = 1$  and is a conservative coefficient of friction estimate between rubber fingerpads and a solid object [48].

From these constraints, which are guided by the mechanics of manipulation, we solve for the equilibrated joint configuration,  $q^*$ , and contact forces on each finger,  $f_N^i$  and  $f_T^i$ , by solving the optimization problem,

$$(q^*, f_N^i, f_T^i) = \operatorname{argmin}_q U(q) \text{ s.t. } (2.18) - (2.26) \quad (2.27)$$

## 2.4.2 Mode Characterization in Simulation

Following these formulations, we create a simulation modeling the motion of an object given an actuation input. Although our simulation can represent any underactuated two-fingered hand-object variant, we decide to limit the parameter variation to just three characteristics in order to maintain tractability of the results. Specifically, we begin with a symmetric two-fingered hand (base variant) and sequentially change link lengths of the right finger, joint stiffnesses of the right finger, and object diameters, while keeping object contact locations constant. To avoid highly asymmetric cases, we choose to incorporate a variational term,  $\Delta l$ , where, if this term is added to one link, it is conversely subtracted from the other in

order to maintain unit length of the finger (Fig. 2.4).

We collect observations of the hand-object system when actuated and save two feature sets of its state. Particularly, these feature vectors consist of both types of finger manipulability measures in addition to the joint configurations of the hand. Concretely, we represent these as feature sets,  $s_m = (w^1, w^2, w_p^2, w_p^2)$  and  $s_q = (q_1^1, q_2^1, q_1^2, q_2^2)$ , which are then both tagged with a mode of manipulation, as determined by the results of the optimization process:

1. Drop - The hand is unable to provide force closure (i.e., when frictional fingertip contacts can equilibrate an external wrench perturbation) on an object of 20 grams, with gravity pointing into the manipulation plane.
2. Stuck - The object is no longer able to move in the direction dictated by actuation forces, creating an excessively large internal object force. This normally occurs at joint limits.
3. Sliding - The object exhibits sliding contacts when normal forces lie outside of the friction cone, as determined by  $\mu_o$  and fingertip forces from (2.27).
4. Normal - The object is manipulable within the gripper’s workspace and modes 1-3 are not satisfied.

We complete the simulation with a total of 867 hand-object variants. Each variant is actuated with a total of 900 distinct actuation pairs, and from each pair, the two feature vectors and mode of manipulation is recorded. Table 2.2 provides a summary of the simulated hand parameters.

Table 2.2: SIMULATED HAND PARAMETERS

Symbol	Value	Symbol	Value
$l_1^1$	6cm	$\Delta l$	[-2.4 – 2.4]cm (17 total)
$l_2^1$	4cm	$d_o$	[2.0 – 6.0]cm (17 total)
$k_2^1/k_1^1$	2.0	$k_2^2/k_1^2$	[2, 2.5, 3] (3 total)
$d_p$	6cm	$r_1^i/r_2^i$	1.2

### 2.4.3 Bounding Feature Distributions by Statistical Testing

The goal of this simulation is to provide general bounds by which grasp mechanics-based features are able to better generalize to different hands compared to joint-based features. More specifically, we approach this study by analyzing the data distributions of each of those feature sets with respect to the modes realized within those distributions, while varying hand-object parameters. It further follows that if mode distributions do not greatly change between hand variants, we are likely able to better transfer learned models to other hands without retuning or retraining with new data.

We perform this analysis by sequentially conducting a One-Way Multivariate Analysis of Variance (MANOVA) test while increasing the difference between tested hand variants. Due to this sequential testing, we adjust the p-value required to reject the null hypothesis according to the Bonferroni correction method, starting with a value of 0.05 at the first variation of testing. This analysis is conducted as follows: given an object diameter and a right finger stiffness ratio, we select three hand variants—two with  $\pm\Delta l$ , and one where  $\Delta l$  is equal to zero (the base variant). We run MANOVA and according to the p-value, decide whether to reject the null hypothesis. For this type of statistical analysis, the null hypothesis tests whether the mode data from the three hand variants come from the same distributions. If the p-value is less than the Bonferroni adjusted threshold, we can reject the null hypothesis, meaning that there is sufficient evidence that the three hand variants do not come from the same data distributions. Performing these tests for all simulated hand-object variants, we compare the p-values of both the mechanics-based feature set,  $\mathfrak{s}_m$ , and the joint-based feature set,  $\mathfrak{s}_q$ . The results are presented in Fig. 2.5.

These results indicate the general bounds by which hand properties can change without varying the feature data distributions for each of the four modes of manipulation. More intuitively, Fig. 2.5 shows that, when starting at the base variant (solid black cell), the green cells are able to extend beyond that of the blue cells, where the green cells denote mechanics-based features and the blue cells denote both, position-based and mechanics-based features. Alternatively, red cells indicate hand variants where both null hypotheses were rejected, i.e., neither feature sets can sufficiently represent the base variant’s data. ’

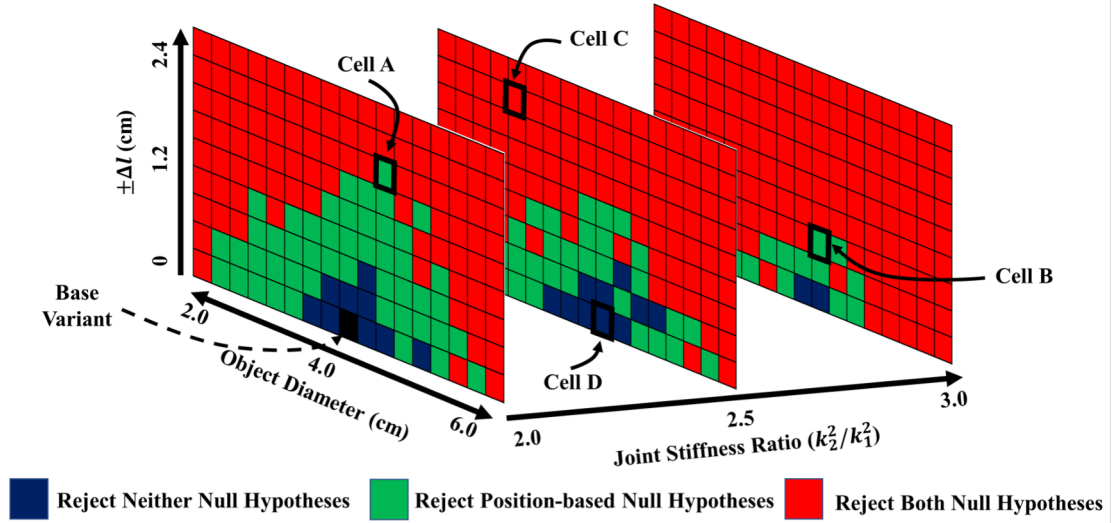


Figure 2.5: MANOVA mode distribution testing of grasp mechanics-based features and position-based features. Green cells generally indicate the extended bounds by which mechanics-based features are able to transfer beyond that of their position-based counterpart (blue cells). Red cells indicate that neither of the feature sets likely share data distributions with the base variant (solid black cell).

In fact, while keeping the object diameter around that of the base variant, and while holding the joint stiffness ratio static,  $\Delta l$  can change by  $\pm 1.8\text{cm}$  without being statistically significant from the other data distributions (Cell A in Fig. 2.5). Notably, this variation cannot extend as drastically when the joint stiffness of the finger also changes. For instance, while still rejecting the position-based null hypothesis, we are only able to change  $\Delta l$  by  $\pm 0.6\text{cm}$  comparatively, but this is when we increase the joint-stiffness ratio from 2 to 3 (Cell B). These two cells, interestingly, have similar p-values of 0.07 for  $\mathfrak{d}_m$ .

This depiction serves to broadly represent the extent of the generalization bounds by analyzing various cells, like Cell C where we can reject both null hypotheses, and Cell D where we cannot reject either of the null hypotheses according to their p-values. While this analysis is not definitive in that it does not necessarily directly transfer to more advanced non-linear learned regression models, it provides a general basis for understanding the added utility of the mechanics-based features from a statistical, data-distribution perspective, and we use this concept to motivate the continuation of this work.

## 2.5 Self-Supervised Tagging and Object Reset

Beyond that of statistical evaluation, we seek to test the reliability of mechanics-based features empirically on a physical hand-object system to further analyze their applicability in real-world environments. We employ such experimentation on an underactuated Yale OpenHand Model T42 [49], that is not equipped with joint encoders or tactile sensors at the fingertips. Due to this limitation, we must reformulate the definitions of the four modes of manipulation:

1. Drop - The hand-object configuration is in a state where the object is just about to drop and will drop shortly thereafter the commanded next actuation.
2. Stuck - The object is no longer able to move in the commanded actuation direction due to the hand-object configuration of the gripper, or the joint has reached a physical hard stop.
3. Sliding - The object exhibits a sliding contact with respect to the gripper’s distal link, i.e., mechanical rolling contact conditions are not satisfied.
4. Normal - The object is manipulable within the gripper’s workspace while maintaining a rolling contact, and modes 1-3 are not satisfied.

### 2.5.1 Manipulation Primitives

The Model T42 is underactuated and thus mechanically compliant, which enables passive reconfiguration post-contact and mitigates potential overconstraint as in a fully actuated hand. This compliance is advantageous for manipulation, as it enables the hand to reconfigure with noisy or imprecise control input. Though due to the nature of this mechanism, we cannot control all degrees of freedom of the object simultaneously, but a 2D submanifold of the object’s 3D configuration space. We employ manipulation primitives on the hand by generating an approximated Jacobian for an arbitrary object that relates to the velocity,  $v = [v_x, v_y, v_\theta]^T$ , of the object frame,  $O \in SE(2)$ , to an actuation velocity,  $\dot{a} = [\dot{a}_1, \dot{a}_2]^T$ , all with respect to the base frame,  $B \in SE(2)$  [50]. These primitive actuation sequences

are estimates of the true Jacobian, and are selected according to the commanded Cartesian velocity reference in the  $x$ -direction,  $v_x$ , and in the  $y$ -direction,  $v_y$  (Fig. 2.6).

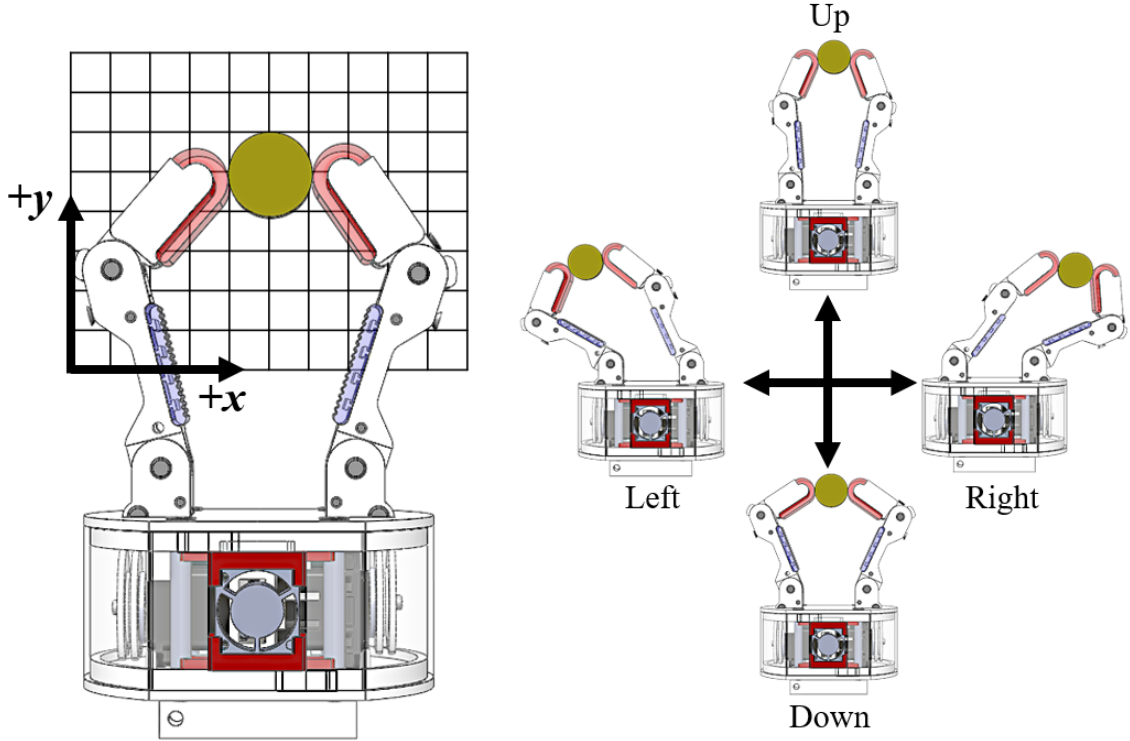


Figure 2.6: Simple manipulation primitives enable planar motion within the workspace of a Yale OpenHand Model T42 gripper. These primitives enable the object to move up, down, left, or right depending on the Cartesian velocity reference,  $v$ .

### 2.5.2 Geometric Hand-Object Representation

The geometric representation of a hand-object system can generally be extracted through various sensing modalities, e.g., cameras, tactile sensors, and joint encoders or IMUs. Albeit, not all hands are equipped with such capabilities, as these types of sensors are generally not required for grasping with compliant hands. Thus, in this work, we focus on a vision-based approach with a fixed overhead camera (30Hz). During manipulation, the gripper configuration is tracked via ArUco markers [51] attached to rigid links of the hand. A priori knowledge of the hand includes the number of finger links, the object geometry, and the geometry of the fingerpads. Principally, the camera pose w.r.t. the task can vary, as long as the markers are visible by the camera sensor such that the 6D pose of the attached markers



can be tracked, e.g., offset from a robot wrist or on another robot.

The state of the contacts is tracked, which subsequently allows the system to detect sliding during manipulation (Sec. 2.5.3), by superimposing a 2D point cloud on both, the fingertips and the object, with respect to the marker frames attached to each. As the hand-object configuration changes during actuation, fundamentally changing characteristics about the grasp such as the effective link length, the superimposed 2D point clouds are tracked and analyzed. We solve for the contact location between the fingerpad and the object by querying a KD-Tree constructed with the object’s point cloud.

### 2.5.3 Self-Supervised Mode Detection

All four modes described in this work can be detected solely by an overhead camera that monitors the hand-object state during manipulation. This observation forms the basis of our self-supervised learning approach, where we can monitor features of the hand and of the object to determine and autonomously tag the current mode of manipulation.

1. *Detecting Drops:* Drop detection is achieved by recording the state history of the object during manipulation. Simply, if the object marker is no longer within the manipulation plane, or the marker is currently absent from visual detection, the object is declared to be dropped. To reduce the potential for drop detection error, the history over the past 10 frames (0.3 seconds or two hand actions) is used to determine such occurrences, whereas this threshold is tuned heuristically during the experimentation setup. If this condition is satisfied, the system accesses the recorded state of the gripper 10 frames prior (directly before the object was dropped) and self-tags a drop observation. The object is then reset via an object reset system (further described in Sec. 2.5.4) and manipulation continues.
2. *Detecting Stuck:* The object is considered stuck if it is no longer manipulable in the direction desired, which is determined by the current Cartesian velocity reference. Typically, this mode occurs when both fingers reach hard stops, limiting additional manipulation towards the palm (Fig. 2.2). Alternatively, stuck cases are also detected when the current configuration of the hand-object system is not able to reconfigure,

limiting the movement of the object in the reference direction. When stuck is detected, the system self-tags an observation and the object is reset via an object reset system (Sec. 2.5.4) for manipulation to continue.

3. *Detecting Sliding:* Sliding is the most difficult of the four modes to detect and is done so when kinematic rolling conditions cannot be satisfied. In order for one surface to be considered rolling on top of another, we choose to track two of the three sliding constraints—the position of the point of contact and the velocity at the point of contact must be the same between the two bodies [52].

Consider the scenario depicted in Fig. 2.7. Here, for the planar case,  $O \in SE(2)$  is the object frame and  $F \in SE(2)$  is the finger frame. To maintain generality,  $F$  can be either finger frame, where the left and right finger frames are  $F^1$  and  $F^2$ , respectively. By parameterizing the object and the fingerpad surface locally in  $O$  and  $F$ , respectively, we effectively develop a point cloud for the object,  $\mathcal{P}_o \in \mathbb{R}^{N \times 2}$ , and for the fingerpad,  $\mathcal{P}_f \in \mathbb{R}^{N \times 2}$ , where the interacting array index from the object point cloud is  $\rho_o$  and the interacting array index for the fingerpad point cloud is  $\rho_f$ , as determined by the KD-Tree. The value for  $N$  can be arbitrarily assigned such that points sufficiently cover the surface of the fingerpad and the object. For clarification, in the object point cloud, the location  $\mathcal{P}_o(\rho_o) \in \mathbb{R}^2$  is in contact with point  $\mathcal{P}_f(\rho_f) \in \mathbb{R}^2$  from the fingerpad point cloud. Let's consider that the location of  $O$  is  $x_o = (O_x, O_y) \in \mathbb{R}^2$ , and the location of the  $F$  is  $x_f = (F_x, F_y) \in \mathbb{R}^2$ , both with respect to  $B$ . We denote the 2D rotation matrices  $R_o$  and  $R_f$  for these respective frames. It follows the elementary consideration that the location of the contact point on the object ( $x_o^c$ ), that is with respect to the base frame, can be calculated as,

$$x_o^c = x_o + R_o \mathcal{P}_o(\rho_o) \tag{2.28}$$

and is similarly calculated for the fingerpad, denoted  $x_f^c$ . To satisfy the positional constraint of a rolling contact, within some user-defined threshold,  $\epsilon_p$ , the following

must be valid:

$$x_o^c - \epsilon_p \leq x_f^c \leq x_o^c + \epsilon_p \quad (2.29)$$

The velocity constraint can be similarly constructed, where we can differentiate the two positions,  $x_o$  and  $x_f$ , with respect to time to form  $\dot{x}_o$  and  $\dot{x}_f$ . Since the body rotations are also functions of time, we must also differentiate body rotations of the object and fingertips to form velocity dependent rotation matrices,  $\dot{R}_o$  and  $\dot{R}_f$  (see [52]). We can then solve for the velocity of the object contact about the base frame,  $\dot{x}_o^c$ , by,

$$\dot{x}_o^c = \dot{x}_o + \dot{R}_o \mathcal{P}_o(\rho_o) \quad (2.30)$$

We similarly calculate  $\dot{x}_f^c$ . Given the velocity threshold,  $\epsilon_v$ , we develop our final constraint:

$$\dot{x}_o^c - \epsilon_v \leq \dot{x}_f^c \leq \dot{x}_o^c + \epsilon_v \quad (2.31)$$

Thresholds  $\epsilon_p$  and  $\epsilon_v$  are tuned heuristically according to the frequency of the camera and the accuracy of contact point estimation. If constraints (2.29) or (2.31) do not hold, it further implies that sliding occurred at the contact. Upon detection, the state of the system is self-tagged and manipulation continues without object reset.

#### 2.5.4 Standardizing Object Reset

Collecting training data for dexterous manipulation is a labor-intensive process, as constant monitoring and manual intervention is frequently required to reset the system due to object drops during manipulation, or from other undesired system scenarios, e.g., actuators at torque limits. Moreover, during reset, it is unlikely that a human can completely standardize the initial grasp of the object, as a human placing the object within the grasp may often cause undesired deviations in the initial pose of the object before manipulation. In order

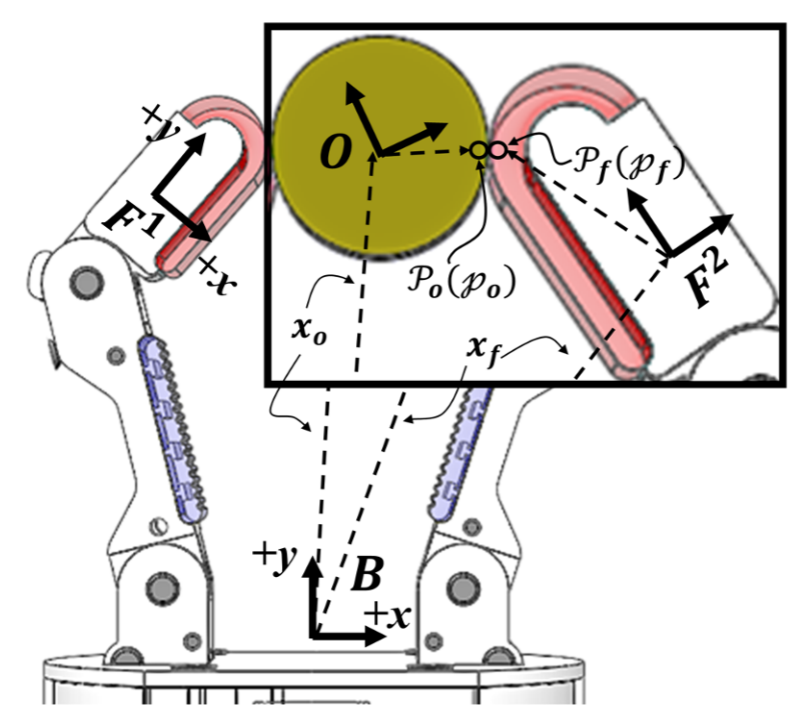


Figure 2.7: Sliding contacts are detected by verifying rolling contact constraints cannot be satisfied. In this depiction, we track the velocity of the hand-object contact points and ensure they are within some normed threshold between each other.

to collect data in a self-supervised manner, we fabricated a system to autonomously and precisely reset the object as to standardize the initial grasp before manipulation.

The automated reset system (Fig. 2.8) is comprised of an object crane and a stabilization beam with an affixed magnet on the end. For each of the objects tested, two magnets were affixed to opposite sides of the body and a lightweight fishing line was strung through the center. During the case of object drop or stuck, the crane raised and the stabilization arm was lowered to the reset position as to adhere to the object magnets. Once the hand reacquired the grasp, which is standardized due to the positioning of the magnets, the stabilization beam lifts out of the way and the crane lowers. This provides slack to the connection between the crane and the object, and allows the hand to freely manipulate the object once again.

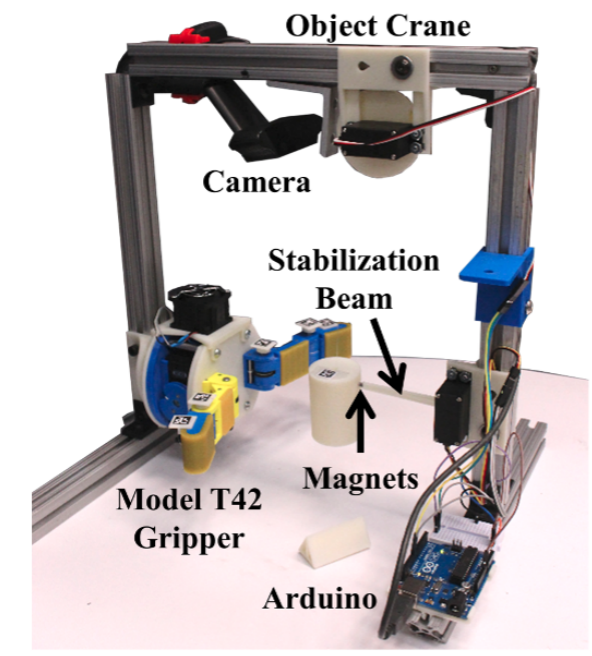


Figure 2.8: An object crane and stabilization beam with affixed magnets accurately resets the object into the same configuration for each trial. This allows us to sequentially collect large amounts of data for training.

## 2.6 Data Collection

We design gripper variants that are generally within the bounds identified in Sec. 2.4 and test the applicability of mechanics-based features empirically, as to evaluate their robustness in physical environments. In a self-supervised manner, we autonomously collect and tag data on 6 gripper variants—one variant for training and five for testing. Once a grasp is acquired after reset, the object was manipulated with randomly selected Cartesian velocity references that operated for a period between 0.5 and 2.5 seconds. A “normal” observation was collected once no other mode was detected for more than 5 seconds. The self-supervised training data was first collected online, randomly selected as to adhere to the leveling of the data distributions, and was then trained and tested offline.

A total of six 3D Printed ABS objects of negligible weight ( $\sim 20\text{g}$ ) and differing geometries in the manipulation plane were created for experimentation (Figs. 2.9, 2.10). The center of each object contained a hole where the object crane was attached. For each object, magnets were affixed to opposite ends as to enable attachment to the stabilization beam for object reset. In the training data, only four objects were used. The other two objects, the oval and the pear, were used as novel test objects.

Training data was collected with a single, symmetric Model T42 gripper variant (PL-PL) with Dynamixel RX-28 actuators. This naming convention signifies a “pivot-long proximal link, and a pivot-long distal link” configuration. From four objects, the two rectangles and the two circles, a total of 3500 modes were collected for training, with an equal mode distribution over each of the objects: 1000 normal, 1000 drop, 1000 stuck, and 500 sliding. Data was tagged and collected until the minimum for each mode was fulfilled. Afterward, overflow mode observations were selected randomly and excluded from the observation set. It is important to note sliding only occurs on objects with flat surfaces, i.e., the rectangular objects (Figs. 2.2, 2.9). Therefore, the number of sliding points recorded for each variant was determined by which type of objects were used during collection. The training data workspace is presented in Fig. 2.11. We note that, generally, the mode regions are symmetric about the central axis of the gripper.

Testing data was collected by equipping the hand with 5 different finger configurations

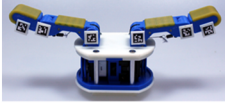

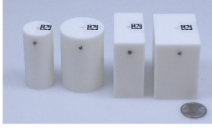
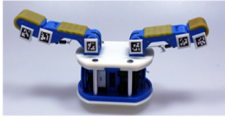

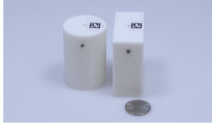


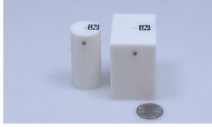


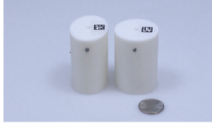


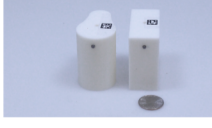
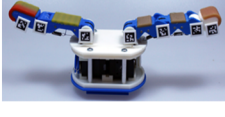


	Variant	Data	Properties	Objects
Training	 	1000 Rolling 1000 Drop 1000 Stuck 500 Sliding	<ul style="list-style-type: none"> <li>• ‘Pivot Long – Pivot Long’</li> <li>• Dynamixel RX-28</li> <li>• Hard Stops: 90°</li> </ul>	
Testing	 	100 Rolling 100 Drop 100 Stuck 50 Sliding	<ul style="list-style-type: none"> <li>• ‘Pivot Long – Pivot Short’</li> <li>• Dynamixel RX-28</li> <li>• Hard Stops: 90°</li> </ul>	
	 	100 Rolling 100 Drop 100 Stuck 50 Sliding	<ul style="list-style-type: none"> <li>• ‘Pivot Short – Pivot Long’</li> <li>• Dynamixel RX-28</li> <li>• Hard Stops: 90°</li> </ul>	
	 	100 Rolling 100 Drop 100 Stuck 0 Sliding*	<ul style="list-style-type: none"> <li>• ‘Pivot Long – Pivot Long Squared’</li> <li>• Dynamixel XM-430</li> <li>• Hard Stops: 90°</li> </ul>	
	 	100 Rolling 100 Drop 100 Stuck 50 Sliding	<ul style="list-style-type: none"> <li>• ‘Pivot Short – Flexure Long’</li> <li>• Dynamixel XM-430</li> <li>• Hard Stops: 70°</li> </ul>	
	 	100 Rolling 100 Drop 100 Stuck 50 Sliding	<ul style="list-style-type: none"> <li>• ‘Pivot Short – Pivot Short – Pivot Short’</li> <li>• Dynamixel XM-430</li> <li>• Hard Stops: 60°</li> </ul>	

Figure 2.9: Manipulation was performed on 6 different gripper variants. The base variant used in training, the symmetric PL-PL gripper, was evaluated with four different objects (small circle, large circle, small rectangle, and large rectangle). A total of 3500 points for training were collected for the four identified modes. The five test variants (PL-PS, PS-PL, PL-PLsq, PS-FL, and PS-PS-PS) then performed manipulation with two of the six test objects. Two novel objects were added in testing (medium oval and medium pear). During manipulation, 50 occurrences of each mode were collected for each gripper-object combination. A quarter is placed next to the objects for size reference. *\*Sliding only occurs with rectangular objects, therefore limiting the number of sliding cases.*

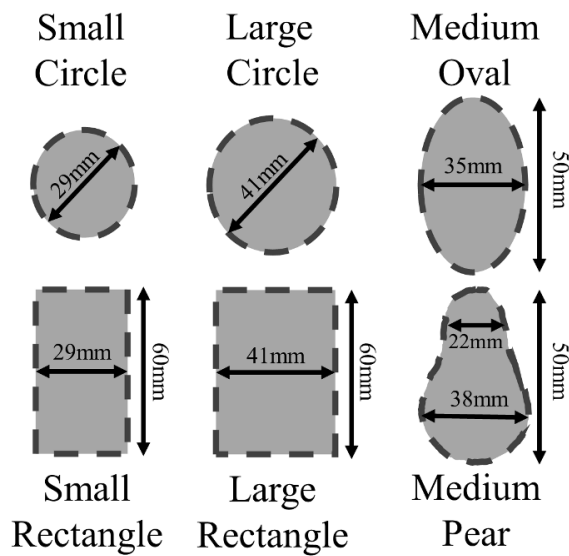


Figure 2.10: Six objects were used for testing and training. In the manipulation plane, object geometries are classified either as a circle, rectangle, oval, or pear.

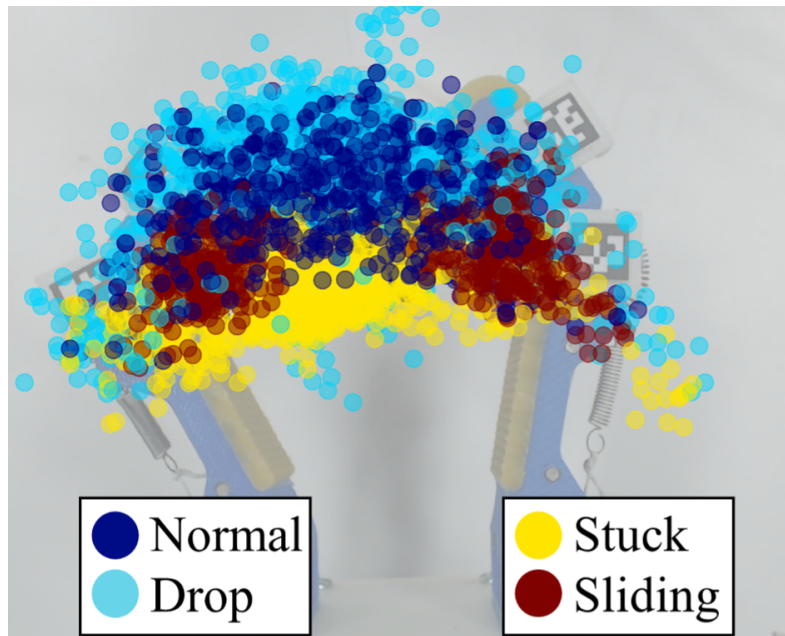


Figure 2.11: Depicting regions of the workspace where modes typically occur. Markers indicate the centroid of the object when a mode was detected. We note the symmetry of this illustration, were modes typically occur mirrored across the central axis of the gripper.



(Fig. 2.9). Variations incorporated changes to the link lengths, fingerpad curvatures, joint types, actuator models, and the number of links compared to the original PL-PL setup. The five variants included PL-PS and PS-PL fingers (Dynamixel RX-28), and PS-FL, PL-PLsq, and PS-PS-PS fingers (Dynamixel XM-430). Different joint stiffness ratios were observed for the PS-FL and PS-PS-PS setups compared to the other four variants. Additionally, the distal hard stop was  $70^\circ$  for the PS-FL variant, and the two hard stops were  $60^\circ$  for the PS-PS-PS variant, compared to all other variants with a distal hard stop of  $90^\circ$ . In each of the five variants used for testing, a total of 50 observations were collected for each mode-object pair. Since in most cases two objects were tested and only one object recorded any sliding, we recorded 100 normal, 100 drop, 100 stuck, and 50 sliding for each variant.

More formally, during data collection we form a feature set,  $\mathcal{S}$ , comprised of features from Sec. 2.3, and a class set,  $\mathcal{R}$ , while manipulating the objects. Denoted by,

$$\mathfrak{s}_n = (v_x, v_y, w^1, w^2, w_p^1, w_p^2, g_{min}, g_{max}, h_{min}, h_{max}, c_f^1, c_o^1, c_f^2, c_o^2) \in \mathbb{R}^{14}$$

an input feature, and

$$r_n = (m) \in \{normal, drop, stuck, slide\}$$

an output feature. The dataset is defined as,

$$\mathcal{S} = \{\mathfrak{s}_n\}_{n=1:M} \quad \mathcal{R} = \{r_n\}_{n=1:M}$$

where its size,  $M$ , has the same number of normal, drop, stuck, and sliding cases for each gripper-object combination.

## 2.7 Experiments

### 2.7.1 Classifier Identification and Observation Reduction

We were first interested in obtaining the best cross validation score given the training feature set,  $(\mathcal{S}_{train} \in \mathbb{R}^{3500 \times 14}, \mathcal{R}_{train} \in \mathbb{R}^{3500})$ , of the symmetric PL-PL setup. In the self-supervised learning approach taken in this work, we evaluated three different predictive models: Random Forest (RF) [53], Support Vector Machines–linear kernel (SVM-l), and Support Vector Machines–radial kernel (SVM-r) [54]. We chose these three classifiers for their extended use in the robotics literature, and due to the fact that other classifiers, e.g., Neural Networks, likely need more data than what was collected in this work. To determine the best classifier for this data, we performed a five-fold cross validation on the training dataset using each classifier. As presented in Table 2.3, the RF classifier performed the best, with an accuracy of 92.3% for all four modes, followed by 88.6% (SVM-r) and 85.4% (SVM-l). For the RF classifier, we calculate a classification accuracy of 85%, 94%, 95%, and 86% for the normal, drop, stuck, and sliding cases, respectively. We note that drop and stuck cases are often classified with higher accuracy than sliding and normal cases. This quality is advantageous as it allows the system to more correctly avoid potentially hazardous modes to stay well within the workspace. For the remainder of this work, we evaluate classification with the RF classifier by building 50 weak learners (shallow trees of depth 10) split according to a Gini impurity measure and averaging each tree’s prediction to determine mode classification.

We were interested in how much data was required to maintain high classification accuracies via self-testing. Using all 14 features from  $s_n$  and the RF classifier, we split the data into two sections: one with 2800 observations (training) and the other with 700 observations (testing), all while keeping the number of modes in each balanced. We continually reduce the number of data points in the training set by 100, removing observations randomly, and test on all 700 test observations. After training the classifier once observations were sequentially removed, we note that the classifier performs similarly with 1200 observations as it does with 3500 observations (accuracy reduces by 3.4%). This 1.7:1 data ratio underscores that sufficient data was collected via self-supervision (Fig. 2.12).

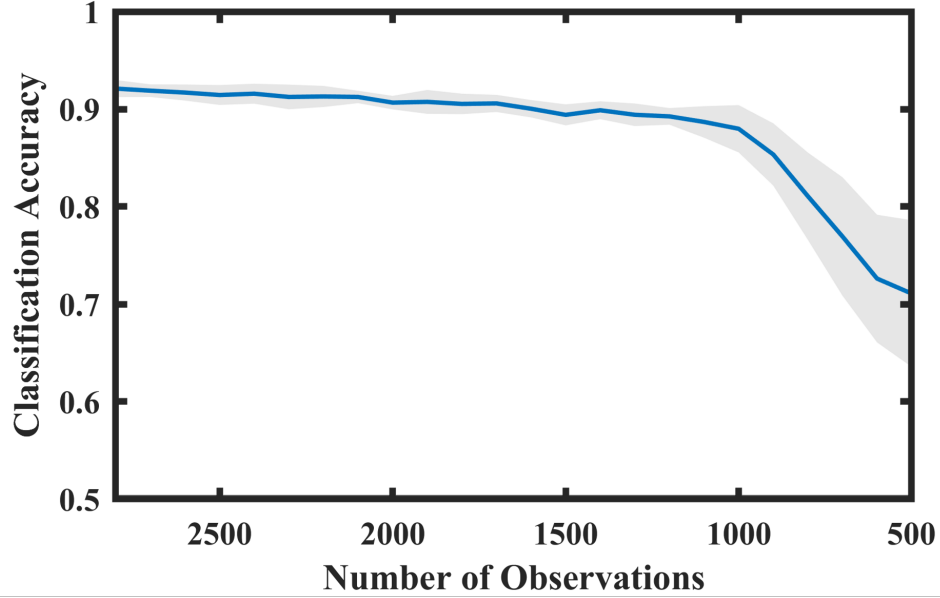


Figure 2.12: Validation of training data size by reducing observations.

Table 2.3: FIVE-FOLD CROSS VALIDATION SCORES ON TRAINING SET

Classifier	Random Forest	SVM-Linear	SVM-Radial
Score	92.3 $\pm$ 0.4%	85.4 $\pm$ 0%	88.6 $\pm$ 0%

## 2.7.2 Classification Accuracy

For the remainder of our analysis, the RF classifier was trained with all 3500 data points using the symmetric PL-PL gripper variant. The particular test set,  $(\mathcal{S}_{test}, \mathcal{R}_{test})$ , was changed according to which of the five gripper variants was being tested.

Using all 14 features from  $\mathfrak{A}_n$ , each of the test grippers were evaluated individually (Fig. 2.13). The classification accuracy of the PL-PS and PL-PLsq variants were highest, with a classification accuracy of 90.6%. The second highest classification accuracy was realized in the PS-PL variant with an accuracy of 85.1%. As provided in the decision matrices in Fig. 2.13 (leftmost column), the PL-PS variant was able to classify normal, drop, and stuck with 84%, 97%, and 99% accuracy, respectively. Classification for sliding dropped to 72%, where it had difficulty distinguishing from the normal mode. The PL-PLsq variant did not have sliding modes, since data was not collected with rectangular objects. Therefore, the lowest classification accuracy was observed with the drop mode (80% accuracy).

This high misclassification of drop is interesting, as it is significantly lower than other variants (97%, 93%, 96%, 88%) with the same feature set. This can be largely attributed to the shifted workspace of the PL-PLsq gripper. As provided by the workspace plots in the rightmost column of Fig. 2.13, compared to the other variants, the modes detected for this variant are shifted to the left of the workspace. Additionally, many drop cases seem to occur in the middle of the workspace, where normal classification would typically be predicted. This artefact is due to the differing geometry of the fingerpad, as it was difficult for the finger to manipulate on the right side of the workspace since the “sharp” edge of the finger prevented a rolling contact to the tip of the finger.

Of the other gripper variants, the 3-link PS-PS-PS performed the worst with a total classification accuracy of 79.8%. In general, for all variants, predicting sliding was difficult as without tactile sensors and with differing friction within the joints of the fingers, the classifier struggles to determine forces applied at the contact point.

### 2.7.3 Feature Reduction

A benefit of the Random Forest classifier is its ability to inherently provide “feature importance measures”, or values that signify how much each feature contributed to the classification decision—providing intuition as to which features were most important during manipulation. In this work, we use a Gini impurity measure to calculate this importance metric, which is a standard often used in ensemble tree classifiers. It works as follows: once a random set of features are selected to determine a split, the Gini impurity represents the likelihood that an accurate classification is predicted given a random class from the distribution of labels. As these splits are calculated for each tree in the forest, the importance measure averages the Gini measures for each split and further signifies the feature’s importance, or more generally, how much “purity” they contributed to the forest. Feature importance measures are reported in Fig. 2.14, where we note that the largest contribution to classification success is attributed to the  $y$ -axis Cartesian velocity reference ( $v_y$ ), finger manipulability measures ( $w^1, w^2$ ), and the penalized finger manipulability measures ( $w_p^1, w_p^2$ ).

Using the feature importance measures, we define 3 feature sets (FS1, FS2, FS3) con-

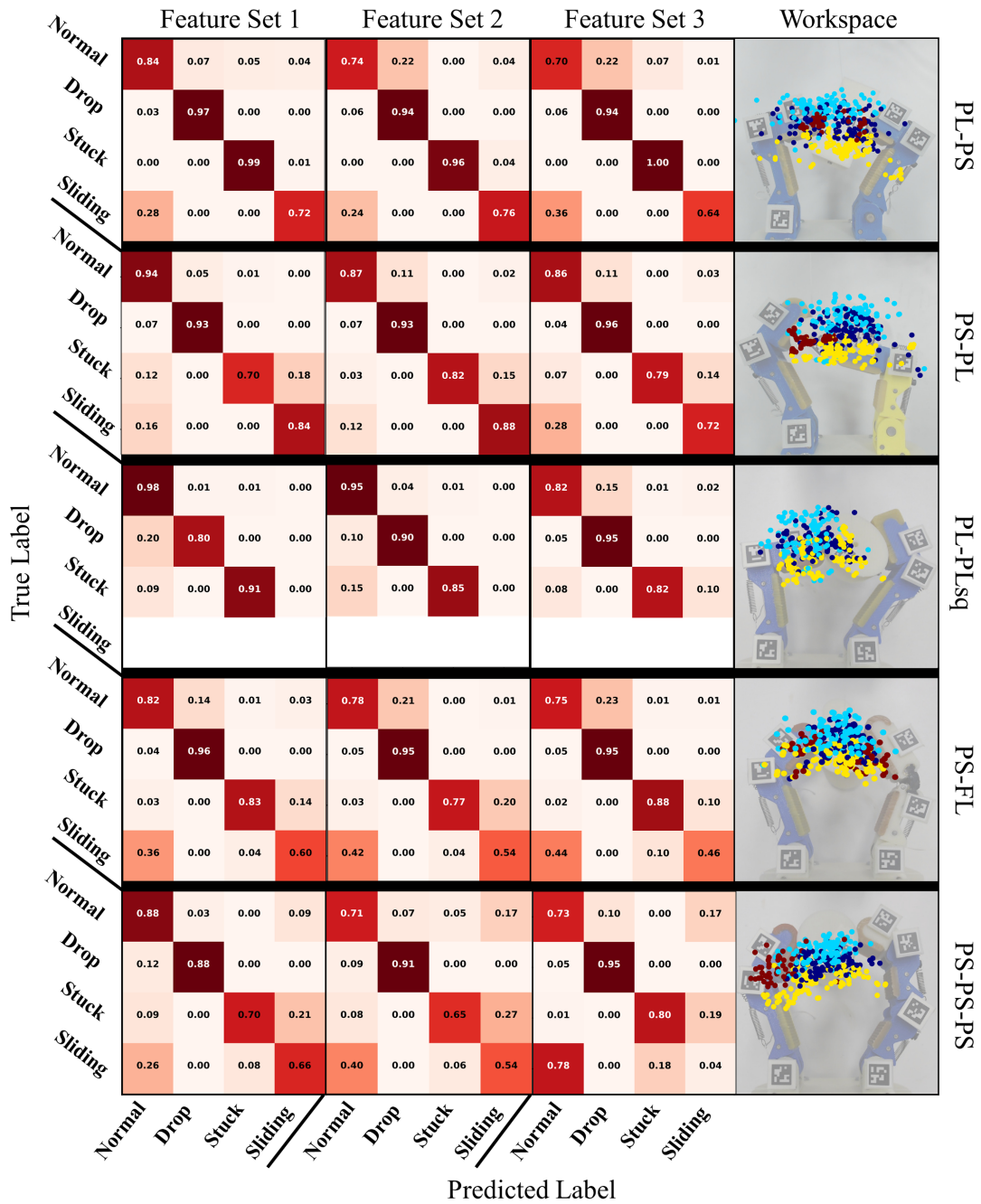


Figure 2.13: (Left three columns) Confusion matrices for each gripper variant given differing feature sets (described in Sec. 2.7.3). (Right column) Object centroid position for modes detected within the workspace of each gripper variant. (Light Blue-Drop, Dark Blue-Normal, Yellow-Stuck, Red-Sliding)

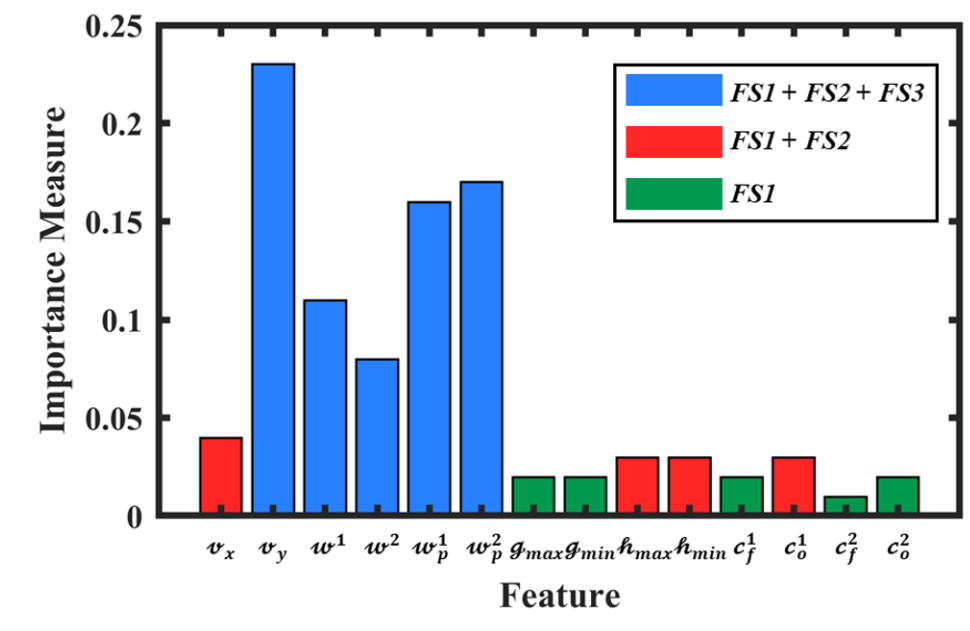


Figure 2.14: Feature importance measures provided by the Random Forest Classifier via Gini impurity. Features in blue are included in Feature Sets 1,2,3, features in red are included in Feature Sets 1,2, and features in green are included in Feature Set 1. See Table 2.4.

sisting of 14, 9, and 5 features, respectively, to further test classification (Table 2.4). By testing accuracy with each feature set, we can provide greater intuition as to what features were most important. The results from this analysis are reported in Table 2.5. and Fig. 2.13. Note that the results from FS1 were previously described in Sec. 2.7.2.

After feature reduction, some variants such as PS-PL (FS1: 85.1%, FS2: 85.0%, FS3:84.4%) and the PL-PLsq (FS1: 90.6%, FS2: 89.2%, FS3: 87.6%) provided consistent classification scores even with the reduction of features. Interestingly, the PS-PS-PS variant obtained nearly the same accuracy between FS1 (79.8%) and FS2 (79.0%), even with the reduction of 5 total features (14 features to 9 features). The PS-FL sees a sharp decrease in classification from FS1 to FS2, but then maintains a similar classification accuracy for FS3. What is also interesting to note, while the overall accuracy of the PL-PLsq decreases, the drop accuracy increases with the reduction of features (FS1: 80%, FS2: 90%, FS3: 95%).

Of the five gripper variants tested, the PL-PLsq maintained the best classification score. This success is likely attributed to two things. First, this variant, dimensionally, is the

Table 2.4: FEATURE SETS DETERMINED BY FEATURE REDUCTION

	Feature Vector
<i>Feature Set 1 (FS1)</i>	$\delta_n = (v_x, v_y, w^1, w^2, w_p^1, w_p^2, g_{min}, g_{max}, h_{min}, h_{max}, c_f^1, c_o^1, c_f^2, c_o^2)$
<i>Feature Set 2 (FS2)</i>	$\delta_n = (v_x, v_y, w^1, w^2, w_p^1, w_p^2, h_{min}, h_{max}, c_o^1)$
<i>Feature Set 3 (FS3)</i>	$\delta_n = (v_y, w^1, w^2, w_p^1, w_p^2)$

Table 2.5: CLASSIFICATION ACCURACY WITH DIFFERING FEATURE SETS

Variant	Feature Set 1	Feature Set 2	Feature Set 3
PL-PS	90.6 $\pm$ 1.2%	87.9 $\pm$ 0.9%	84.1 $\pm$ 1.6%
PS-PL	85.1 $\pm$ 0.9%	85.0 $\pm$ 1.1%	84.4 $\pm$ 2.1%
PL-PLsq	90.6 $\pm$ 2.2%	89.2 $\pm$ 1.2%	87.6 $\pm$ 1.4%
PS-FL	84.8 $\pm$ 1.8%	78.3 $\pm$ 2.4%	77.6 $\pm$ 1.6%
PS-PS-PS	79.8 $\pm$ 0.7%	79.0 $\pm$ 1.3%	70.0 $\pm$ 0.8%

closest variant to the original PL-PL used in training, as the only difference is the squared fingertip on the right finger. Second, this variant was tested with two rounded objects (circle and oval), and therefore no sliding occurred during manipulation, which is normally the most difficult to classify. While evaluated variants that were tested with sliding cases, the PS-PL performed the best with a total classification accuracy of 84.4% (FS3). This variant also had the highest sliding accuracy among any of the five variants throughout all features sets, which can likely be attributed to the fact that this variant has the same distal link as the training PL-PL variant. The PS-PS-PS gripper variant performs the worst of the five variants—this variant has a more limited workspace due to the hard stops at 60° at each of the links. Additionally, sliding only occurs on the left side of the workspace, since the flat surface of the right most-distal link rarely comes in contact with the object. As depicted by the confusion matrices in Fig. 2.13, in general, as the number of features is reduced, the ability for the system to accurately predict sliding greatly reduces. For example, in the PS-PS-PS variant for FS1, the classification for sliding is 66%, but in FS3 the accuracy is just 4%. This is a fairly specific case, as the classification accuracy for sliding only differs from a maximum of 12% for the three other gripper variants (from FS1 to FS3).

As previously discussed, the rightmost column of Fig. 2.13 provides workspace plots for the 5 test gripper variants. Plotted points depict the centroid of the object when a mode

was detected. While tested objects were of various geometries, this plot generally presents where modes were likely to occur within the workspace. It is interesting to note how the “regions” for different modes change according to the gripper variant, especially how varied they are compared to the training hand, PL-PL, in Fig. 2.11. For example, sliding only occurred on the left side of the workspace for the PS-PL and PS-PS-PS variants, and a large number of drop cases occurred in the middle of the workspace for the PS-FL variant. These workspace plots underscore how, where the joint configuration and object center location inside of the workspace is important, the properties of the hand-object system must be accounted for in order to accurately predict modes of manipulation.

#### **2.7.4 Single-Component Feature Reduction**

As stated in Sec. 2.7.2, some variants were not as susceptible to higher classification errors given feature reduction techniques, while others were more affected. It was our interest to perform feature reduction techniques by removing one feature at a time, instead of in sets, as to validate our approach. We begin by removing features from least important to most important according to the measures presented in Fig. 2.13.

The results to this feature reduction are presented in Fig. 2.15. While performing this task on the PL-PL variant with a total of 3500 observations, we note that the cross-validation accuracy remains around 93% while having 9 or more features. Thereafter, when only 8 features remain, the accuracy drops to 87% and continues until 5 features remain. Once only 4 features are used for classification, the accuracy starts to decline, as it is difficult to determine the decision boundary. This feature reduction test validates the decision for 14, 9, and 5 features for FS1, FS2, and FS3, respectively (Sec. 2.7.3), as these are volatile intervals when accuracy will likely drop.

#### **2.7.5 Online Classification**

Detection of modes, and their associated regions, are somewhat fluid (as presented by the workspace plots) and in general, we are interested to see if modes can be successfully predicted online to promote safe manipulation. We implemented this RF prediction model in an online framework to evaluate classification accuracy with two novel gripper variants not



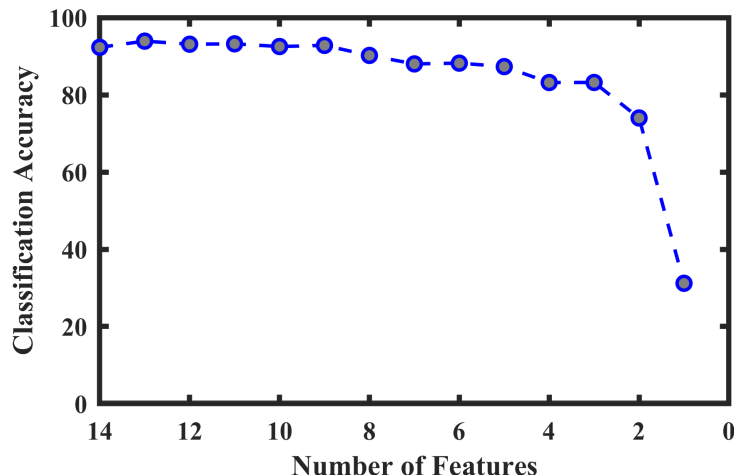


Figure 2.15: Five-fold cross validation accuracy of the PL-PL training variant. Features were reduced one at a time subject to their classification accuracy contribution (see Fig. 2.14).

evaluated in the previous sections (Fig. 2.16). The first variant was tested with the medium oval and consisted of a PS-FL left finger and a PS-PS-PS right finger. The second variant was tested with the small rectangle and was comprised of a PS-FL left finger and a PL-PS right finger. As before, the mechanics-based features used for testing were extracted online using markers attached to rigid links of the hand. The gripper was commanded through random Cartesian velocity references for a period between 0.5-4.0 seconds to attempt to cover the entire workspace. Once a mode other than normal was detected for a period between 0.1-1.0 seconds, the Cartesian velocity reference changed randomly to either stop manipulation or guide the object back towards the middle of the workspace.

Online classification using the first novel variant properly classified modes normal, drop, and stuck within its workspace for the oval object. In addition to these three modes, the second variant also included the “sliding” mode. To test the efficacy of this online detection, the classifier was run on each hand 5 times for 5 minutes. Cartesian velocity references were selected randomly, with a goal of remaining within the manipulable region of the gripper. For the first variant, 3 out of the 5 executions were successfully run for a total of five minutes. The object was manipulated safely within the workspace and was diverted towards the center of the workspace when a mode other than normal was detected. For the other two executions, a dropped object was detected at 3 minutes 21 seconds and 4 minutes

5 seconds. For the second variant, 4 out of the five executions successfully completed 5 minutes of manipulation. The final failed execution successfully manipulated the object for 2 minutes and 34 seconds. This failure was due to the amount of sliding the object had undergone without detection 12 seconds before task failure.

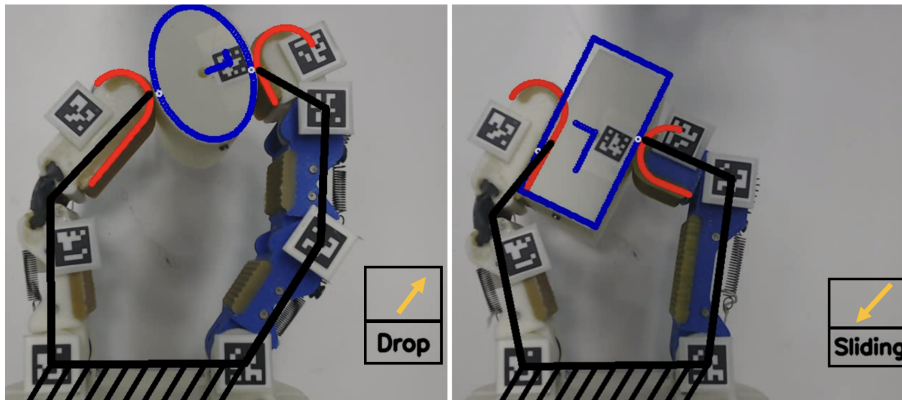


Figure 2.16: Online classification of two novel gripper variants. The arrow signifies the Cartesian velocity reference and the text (Drop or Sliding) signifies the predicted mode. (Left) A PS-FL left finger and a PS-PS-PS right finger perform manipulation and the online classifier predicts a drop will occur given the Cartesian velocity reference. (Right) A PS-FL left finger and a PL-PS right finger predicts sliding will occur during manipulation.

## 2.8 Discussions

In this chapter, we showed that by learning from mechanics-based features, which represent high-level properties of the hand-object system, we were able to successfully transfer mode prediction accuracies between different gripper variants. We first provided bounds by which mechanics-based features were likely to better transfer than their joint-based counterparts. We then tested this notion physically with different hand variants. Specifically, the 92.3% five-fold cross validation accuracy of the training variant was marginally greater than the 90.6% classification accuracy of the PL-PLsq and PL-PS variants. However, we did note that these features presented shortcomings in accurately predicting sliding when stiffness ratios changed between variants. Additional sensing modalities such as tactile sensing at the fingertips would likely be beneficial for classification.

The features included in FS3, or the set containing finger manipulability measures, con-

tribute the most to the success of the classifier (75.2% of the Gini impurity measure). The  $v_y$  component benefits classification in that, according to the workspace plots, it likely discriminates between the drop, normal, and stuck regions as the  $y$ -position component passes through all three. When coupled with the finger manipulability measures, these values together determine where the object is within the workspace and where it is headed, and in general, the hand-object configuration. It is our belief that the other features defined, such as singular values of the Hand-Object Jacobian and the contact curvatures, are important for stable manipulation capabilities when fingerpad curvatures change more drastically, or different gripper types (underactuated vs. fully actuated) are observed.

This work elucidates the beginning of what we consider a promising approach for learning models in dexterous manipulation. While we recognize the drawbacks and inaccuracies in predicting sliding as the gripper becomes more asymmetric, this approach has proven to be successful for the other three modes, and was completed without the use of tactile sensing. Although conceptually backed by simulation, the majority of our analysis consisted of data that was collected physically, which allows us to capture uncertainties of the real world. In future work, we plan to investigate this approach further by extending this sort of classification to the spatial manipulation case, investigating how time series data aids in prediction accuracy (e.g., HMMs), further modeling this approach for deformable contacts and objects, and testing such methods on more commercial, readily-available robot grippers. Furthermore, we are interested in how adding single unit tactile sensors at the fingertips may be beneficial in detecting sliding cases when using different gripper variants.

While this approach of using mechanics-based features for learning dexterous manipulation can be applied to any hand design, it is particularly useful for soft, compliant, or underactuated hands that typically do not have tactile sensors or joint encoders. Fundamentally, dexterous manipulation extends the workspace of the manipulator and is a valuable tool for the future of robotics in society. We hope that the robustness demonstrated by testing different gripper variants encourages researchers to search for features that represent higher-level properties of the system for a more enlightened discussion on learning system models.

## Chapter 3

# Controlling a Compliant Hand for In-hand Manipulation

### 3.1 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 [21]. 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.

In this chapter, 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  $SE(3)$ .

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 chapter 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. In the continuation of this chapter please refer to Table 3.1 for nomenclature.

Table 3.1: CHAPTER 3 NOMENCLATURE

<b>Symbol</b>	<b>Description</b>
$q$	Particular hand configuration: $q \in \mathbb{R}^6$
$a$	Configuration of the actuators: $a \in \mathbb{R}^3$ where $\dot{a}$ denotes velocity
$E(q)$	Potential energy of the system in a specific joint configuration $q$
$\mathcal{X}$	Pose of the Grasp Frame: $SE(3)$
$\mathcal{T}$	Contact triangle relationship: $(\mathcal{T}_1, \mathcal{T}_2, \mathcal{T}_3) \in \mathbb{R}^3$

## 3.2 Related Work

### Analytical Modeling for Manipulation

Many works have approached dexterous manipulation with various levels of analytical modeling—from contact models [55] and fingerpad curvature models [47], to hand kinematic models [56] and whole hand-object system models [21]. 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 [18]. Some of these problems are nullified when using underactuated, adaptive hands that inherently reconfigure to uncertainties such as noisy control inputs or modeling errors [57]. Nevertheless, dexterous manipulation with such hands remains difficult to model as the output space is typically of higher dimension than the input space.

### Learning for Manipulation

To overcome uncertainties in the analytical models, learning for manipulation—both model-based [58], [38] and model-free approaches [59]—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 [60]. 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 [61], [62] or through tactile sensing [63]. These works, however, focus mainly on the motion of the object/grasp frame and not on a generalized manipulation frame attached to the object.

### Control for Manipulation

Control for manipulation has been similarly approached from various avenues—with methods based purely on kinematics [64], tactile sensing [65], and visual servoing [66], [67]. It is also possible to combine sensing modalities for additional control, e.g., for grasp adap-

tation [68]. 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 [69], 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.

### 3.3 Devising a Manipulation Model

In this section, we present an approach to learning the manipulation model of an underactuated hand through an energy-based perspective [61]. 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 [57], [69].

#### 3.3.1 The Grasp Frame

The establishment of the grasp frame generalizes the geometric properties of an arbitrary object within a grasp [70]. Fundamentally, it portrays the local geometry of the object and standardizes the representation of the object frame (Fig. 3.1, 3.2). 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  $P = p_1, \dots, p_k$  where  $p_i \in \mathbb{R}^3, \forall i \in \{1, \dots, k\}$  with respect to the hand frame. Noteworthy, 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,

$$\begin{aligned}
\mathcal{X} &= [\mathcal{G}_x, \mathcal{G}_y, \mathcal{G}_z | \mathcal{G}_o] \in SE(3) \\
\mathcal{G}_o &= \frac{1}{3}(p_1 + p_2 + p_3) \\
\mathcal{G}_x &= \frac{p_2 - p_1}{\|p_2 - p_1\|_2} \\
\mathcal{G}_z &= \frac{(p_3 - p_2) \times \mathcal{G}_x}{\|(p_3 - p_2) \times \mathcal{G}_x\|_2} \\
\mathcal{G}_y &= \mathcal{G}_z \times \mathcal{G}_x
\end{aligned} \tag{3.1}$$

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,

$$\mathcal{T} = (\|p_1 - p_2\|_2, \|p_2 - p_3\|_2, \|p_3 - p_1\|_2) \in \mathbb{R}^3 \tag{3.2}$$

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 object dynamics. Additional generalization of object dynamics will be addressed in future work. Note that this energy model follows a similar idea to that as described in Chapter

### 3.3.2 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,

$$r_{ai}\dot{a}_i = r_{pi}\dot{q}_{pi} + r_{di}\dot{q}_{di} \tag{3.3}$$



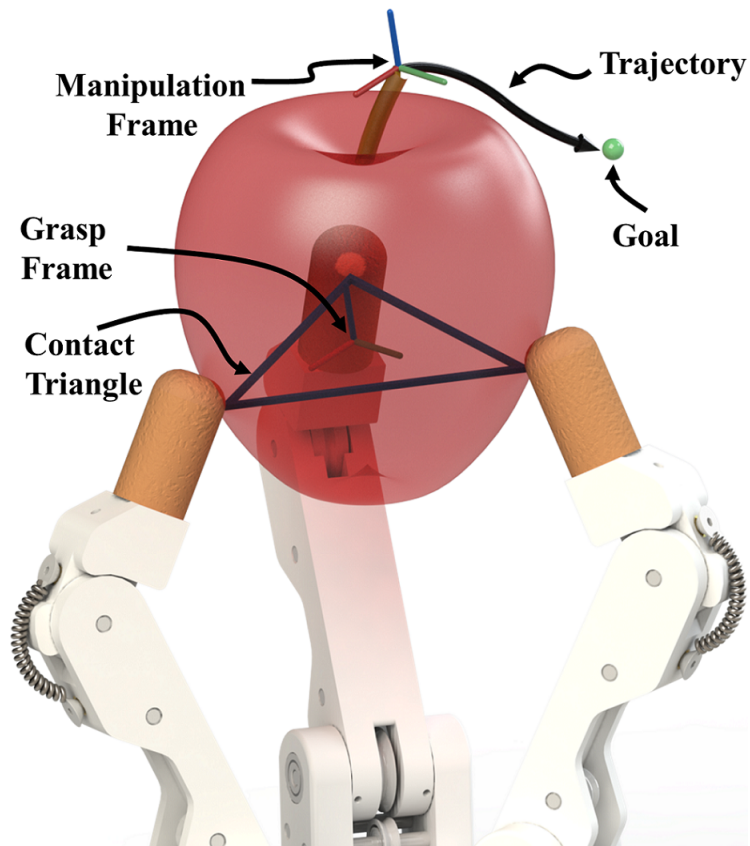


Figure 3.1: Partially constrained trajectories of the manipulation frame, e.g.,  $\in \mathbb{R}^3$ , leave uncertainties in grasp frame planning since the mobility of the mechanism is subject to constraints imposed by the closed kinematic chain. The proposed framework utilizes Model Predictive Control to solve for a valid grasp frame trajectory with any underconstrained reference.

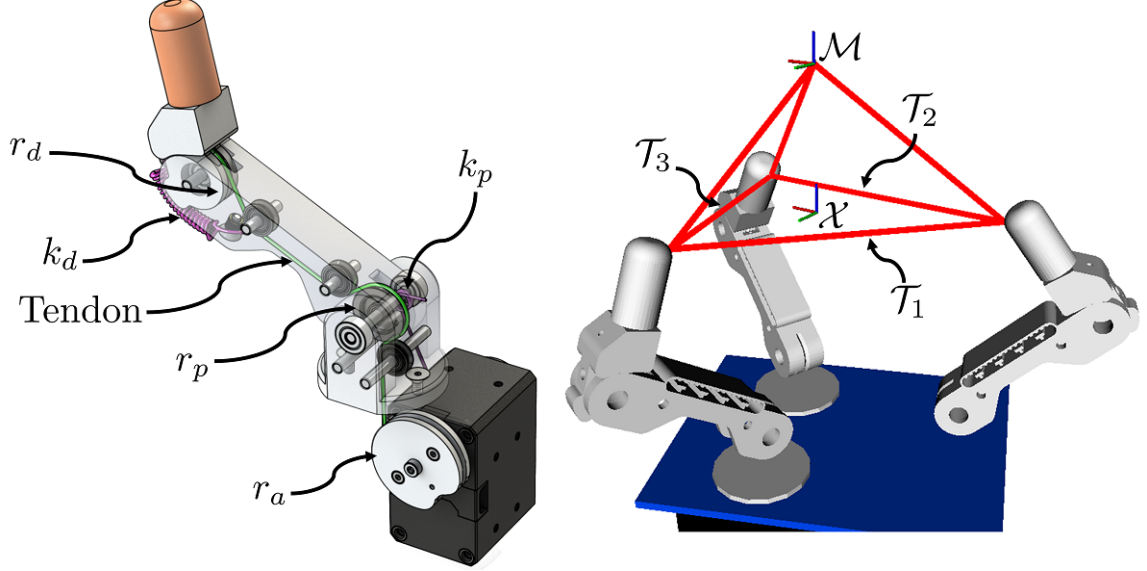


Figure 3.2: (Left) The tendon transmission of an underactuated finger is dependent on pulley and spring parameters. (Right) Object geometry can be generalized by evaluating the triangle relationship,  $\mathcal{T}$ , between the contacts, and offsetting the manipulation frame,  $\mathcal{M}$ , from the grasp frame,  $\mathcal{X}$ .

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,

$$q^* = \underset{i}{\operatorname{argmin}} \sum E^i(q^i) \quad \text{s.t. (3.2), (3.3)} \quad (3.4)$$

where  $E^i$  is the potential energy of the  $i^{\text{th}}$  finger,

$$E^i(q^i) = \frac{1}{2}(k_p q_{pi}^2 + k_d q_{di}^2) \quad (3.5)$$

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. 3.2). 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 [61]. 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,

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

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

$$g : (\mathcal{X}, \dot{\mathcal{X}}, \mathcal{T}) \rightarrow \dot{a} \quad (3.6)$$

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.

Note that this energy model follows a similar idea to that as described in Chapter 2, but enacts fewer constraints onto the system as to maintain generality. Particularly, the grasp frame representation enables an object-agnostic learning style to manipulation.

### 3.4 Controlling In-Hand Manipulation

For the continuation of this work, the main control algorithm is illustrated in Fig. 3.3 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)

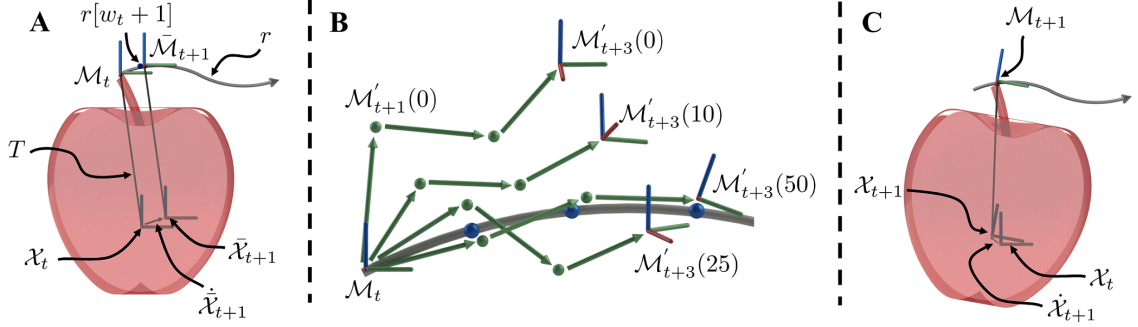


Figure 3.3: A.) The manipulation frame,  $\mathcal{M}_t$ , can be represented by a rigid transformation,  $T$ , from the grasp frame,  $\mathcal{X}_t$ . In Alg. 2 a bidirectional guess initializes the model’s input variables by assuming that the next grasp frame pose,  $\bar{\mathcal{X}}_{t+1}$ , has 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  $\mathcal{M}_{t+1}$ .

### 3.4.1 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  $J$ ,

$$J = \sum_{i=1}^{k_c} \gamma \|r[w_{t+i}] - \mathcal{M}_{c,t+i}\|_2 + \dots \tag{3.7}$$

$$\sigma \|r_{end} - \mathcal{M}_{c,t+i}\|_2 + \lambda \|\dot{a}_{t+1}\|_2$$

where  $\gamma$ ,  $\sigma$ , and  $\lambda$  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  $\sigma$  and decreasing  $\gamma$  and  $\lambda$  will do this with the trade-off of likely decreasing trajectory accuracy.

### 3.4.2 The Manipulation Controller

Using this cost-minimization approach, we formulate the control process as illustrated in Fig. 3.3 and as outlined in Alg. 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 (3.7) that can be used to compare the utility of each trajectory.

#### 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

---

**Algorithm 1** MPC with Stochastic Hill Climbing Optimization
 

---

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

```

1:  $\mathcal{C}_{best} \leftarrow Trajectory()$  ▷ initialize first trajectory
2:  $\mathcal{C}_{best}.addNode(\mathcal{X}_t, \dot{\mathcal{X}}_0 = 0, \dot{a}_0 = 0)$  ▷ start node
3: for  $t = 1$  to  $k_p$  do ▷ prediction horizon
4:    $\bar{\mathcal{X}}_{t+1} \leftarrow bidirInit(\mathcal{C}_{best}.n[t].\mathcal{X}, r, c)$  ▷ Alg. 2
5:    $\dot{a}_{t+1} \leftarrow g : (\mathcal{C}_{best}.n[t].\mathcal{X}, \bar{\mathcal{X}}_{t+1}, \mathcal{T})$  ▷ (3.6)
6:    $\mathcal{X}'_{t+1}(0) \leftarrow Hand.evaluate(\dot{a}_{t+1})$  ▷ (3.4)
7:    $\dot{\mathcal{X}}'_{t+1}(0) \leftarrow diff(\mathcal{C}_{best}.n[t].\mathcal{X}, \mathcal{X}'_{t+1}(0))$ 
8:    $\mathcal{C}_{best}.addNode(\mathcal{X}'_{t+1}(0), \dot{\mathcal{X}}'_{t+1}(0), \dot{a}_{t+1})$ 
9:    $\mathcal{M}_{t+1}(0) \leftarrow Hand.manipFrame(\mathcal{X}'_{t+1})$ 
10:  if  $\|\mathcal{M}_{c,t+1}(0) - r_{end}\|_2 < \epsilon$  then
11:    break ▷ reached goal
12:
13: for  $i = 1$  to  $iter$  do ▷ optimization iterations
14:    $\mathcal{C}_i \leftarrow Trajectory()$  ▷ initialize new trajectory
15:    $\mathcal{C}_i.addNode(\mathcal{X}_t, \dot{\mathcal{X}}_0 = 0, \dot{a}_0 = 0)$ 
16:   for  $t = 1$  to  $k_p$  do
17:      $\dot{\mathcal{X}}'_{t+1}(i) \leftarrow perturb(\mathcal{C}_{best}.n[t+1].\dot{\mathcal{X}})$  ▷ Alg. 3
18:      $\dot{a}_{t+1} \leftarrow g : (\mathcal{C}_i.n[t].\mathcal{X}, \dot{\mathcal{X}}'_{t+1}(i), \mathcal{T})$  ▷ (3.6)
19:      $\mathcal{X}'_{t+1}(i) \leftarrow Hand.evaluate(\dot{a}_{t+1})$  ▷ (3.4)
20:      $\mathcal{C}_i.addNode(\mathcal{X}'_{t+1}(i), \dot{\mathcal{X}}'_{t+1}(i), \dot{a}_{t+1})$ 
21:      $\mathcal{M}_{t+1}(i) \leftarrow Hand.manipFrame(\mathcal{X}'_{t+1}(i))$ 
22:     if  $\|\mathcal{M}_{c,t+1}(i) - r_{end}\|_2 < \epsilon$  then
23:       break ▷ reached goal
24:   if  $Cost(\mathcal{C}_i) < Cost(\mathcal{C}_{best})$  then ▷ (3.7)
25:      $\mathcal{C}_{best} = \mathcal{C}_i$  ▷ better trajectory
26:
27: return  $\mathcal{C}_{best}.n[1].\dot{a}$ 

```

---

is outlined in lines 1-11 of Alg. 1 and is depicted in Fig. 3.3.A.

To formulate the first trajectory, we rely on a bidirectional initialization presented in Alg. 2. This procedure initializes a first guess for the grasp frame velocity,  $\bar{\mathcal{X}}_{t+1}$ , 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,  $\bar{\dot{\mathcal{X}}}_{t+1}$ , is finally estimated by taking the element-wise difference between  $\mathcal{X}_t$  and  $\bar{\mathcal{X}}_{t+1}$ .

After the bidirectional initialization guess,  $\bar{\dot{\mathcal{X}}}_{t+1}$  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,  $\epsilon$ .

### 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 (3.7). Here, *iter* represents the number of optimization iterations we intend to compute. This process is depicted in Fig. 3.3.B and references lines 13-25 of Alg. 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 (Alg. 3). The learned model then evaluates this grasp

---

**Algorithm 2** *bidirInit*( $\cdot$ )

---

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

```
1:  $\mathcal{M}_t \leftarrow \text{Hand.manipFrame}(\mathcal{X}_t)$ 
2:  $w_t \leftarrow \text{nearestWaypoint}(\mathcal{M}_{c,t}, r)$ 
3: for  $l$  in  $[x, y, z, \theta_R, \theta_P, \theta_Y]$  do
4:   if  $l \subseteq c$  then
5:      $\bar{\mathcal{M}}_{l,t+1} \leftarrow r[l, w_t + 1]$ 
6:   else
7:      $\bar{\mathcal{M}}_{l,t+1} \leftarrow \mathcal{M}_{l,t}$ 
8:  $T \leftarrow \text{getTransform}(\mathcal{X}_t, \mathcal{M}_t)$ 
9:  $\bar{\mathcal{X}}_{t+1} \leftarrow \text{applyInvTransform}(\bar{\mathcal{M}}_{t+1}, T)$ 
10:  $\dot{\bar{\mathcal{X}}}_{t+1} \leftarrow \text{diff}(\mathcal{X}_t, \bar{\mathcal{X}}_{t+1})$ 
11: return  $\bar{\mathcal{X}}_{t+1}$ 
```

---

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,  $\epsilon$ , the loop breaks prematurely. Once a trajectory of  $k_p + 1$  in length is computed, we compare 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,  $\mathcal{C}_{best.n[1]}.a$ . This input is then executed physically (Fig. 3.3.C) and results in the actual system transition from  $\mathcal{M}_t$  to  $\mathcal{M}_{t+1}$ , and similarly,  $\mathcal{X}_t$  to  $\mathcal{X}_{t+1}$ . Alg. 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:**  $\dot{\mathcal{X}}'_{t+1}$ 

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

## 3.5 Experiments

The proposed control framework was instantiated on a 3-fingered underactuated Yale Open-hand Model O [49]. 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. 3.2).

The learned model in (3.6) 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 (3.4). 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 [61].

### 3.5.1 Translational Trajectory Control

We implemented translational control, i.e.,  $c = (x, y, z)$ , in a simulated environment (Fig. 3.2) while varying the control horizon and number of optimization iterations as to tune the controller. This test, presented in Fig. 3.4, 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

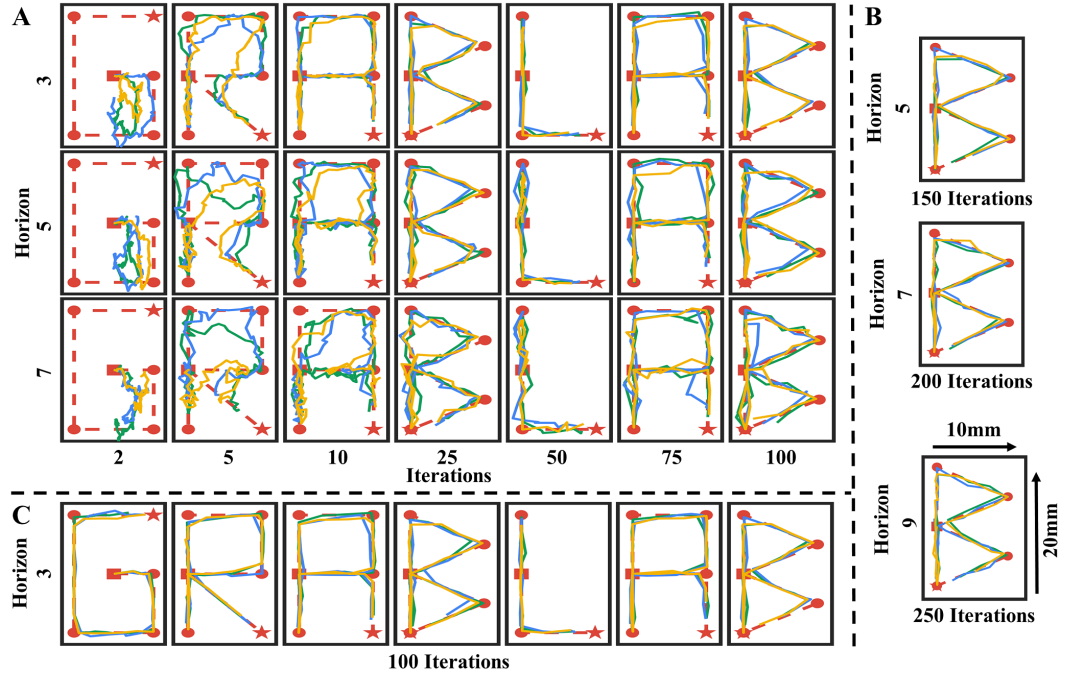


Figure 3.4: Translation control,  $c = (x, y, z)$ , of the manipulation frame depicting the reference trajectory in the  $x - y$  plane (Red), and the trajectories of Obj. 1 (Green), Obj. 2 (Yellow), and Obj. 3 (Blue). A.) We trace the letters 'GRABLAB' while varying control horizons and optimization 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).

Table 3.2: OBJECT PROPERTIES FOR THOSE USED IN SIMULATION

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)

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  $T_p$ .

Table 3.2. Each letter was 20mm in height and 10mm 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. 3.4.A depicts a test correlating accuracy to varying horizon lengths and optimization iterations. Generally, we note that as 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. 3.4.B, where we record similar trajectory errors (0.72mm 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. 3.4.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 2mm is sufficient for legibility. Testing up to 100 iterations (0.45mm error), the results show that 50 iterations (0.95mm error) is sufficient to satisfy the accuracy required by the task, presented in Fig. 3.5. For this reason, we will proceed in the next sections by evaluating trajectories with this configuration.

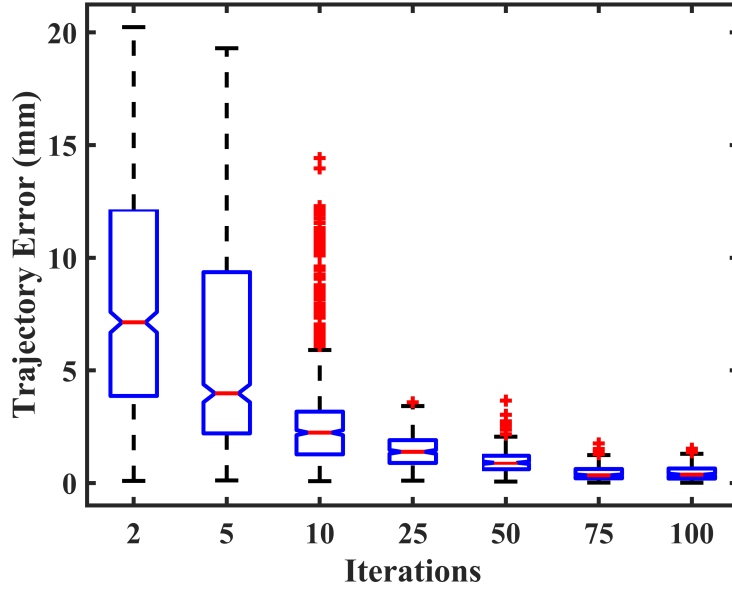


Figure 3.5: With a prediction horizon of 3, the letters 'GRABLAB' were traced with three different objects while varying optimization iterations. The error experienced during execution was recorded for each of the trajectories. We identify an elbow point of 50 iterations satisfies the desired task accuracy.

### 3.5.2 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.

In each of these tests, the hand was initialized with the same hand configuration as in Fig. 3.2, 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. 3.6, 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

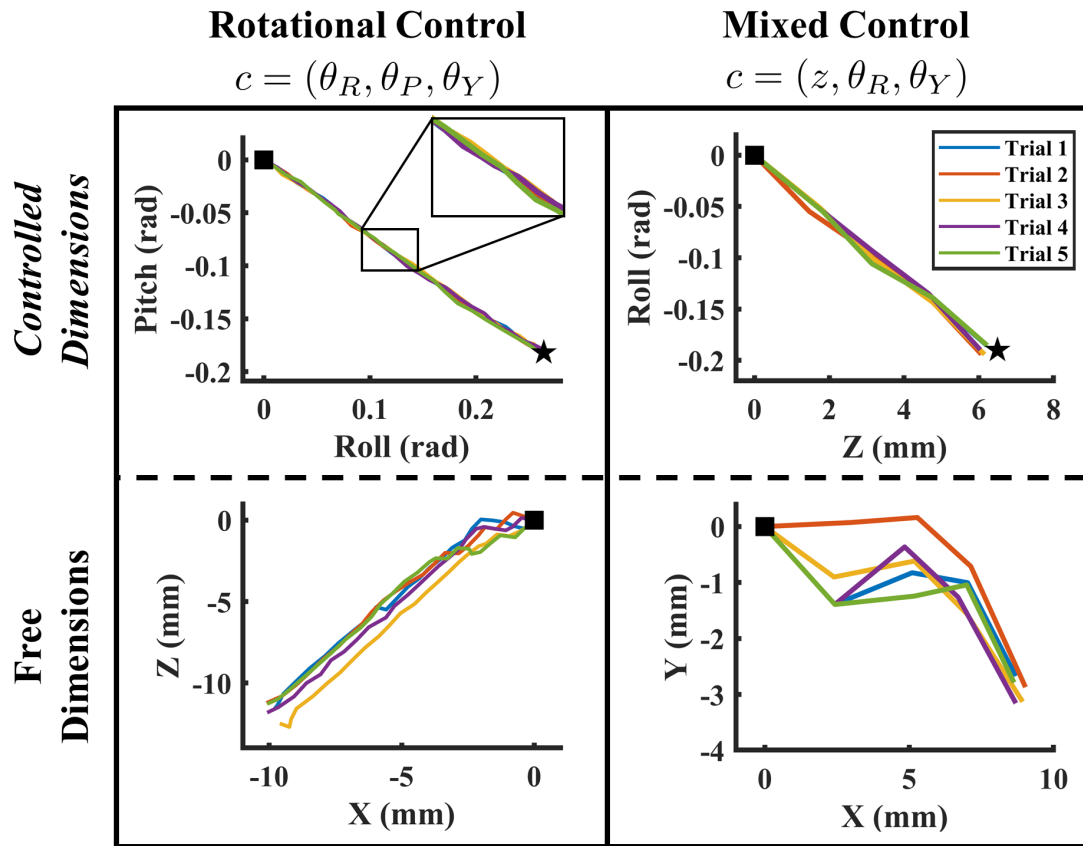


Figure 3.6: A single trajectory in Rotation Control (left) and a single trajectory in Mixed Control (right) was executed for 5 trials. The *controlled dimensions* (top) follow the trajectory as desired. The free dimensions (bottom) are allowed to drift to any trajectory that adheres to the system constraints. The start configuration is denoted with a square and the goal configuration (only in the *controlled dimensions*) is denoted with a star.

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.

### 3.5.3 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) [71]. 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. 3.7, 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 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. 3.8). 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.

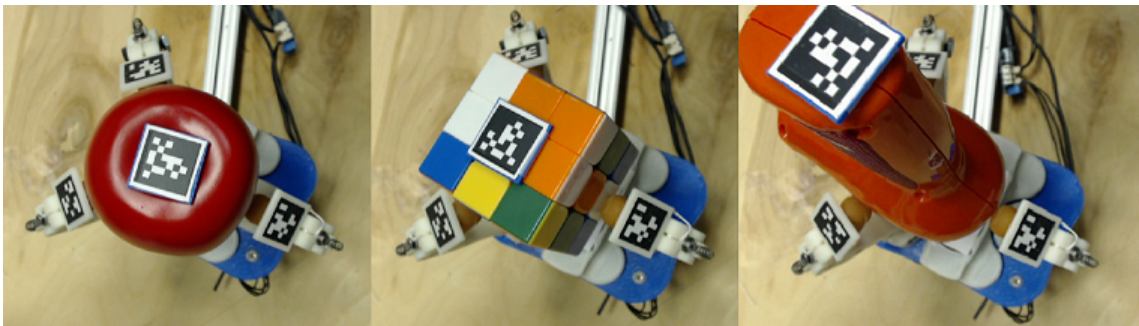


Figure 3.7: Top view of the apple, Rubik's Cube, and drill from the YCB Object and Model Set used for physical testing of the control framework.

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. 3.7). This generated initial contact triangle relationships and

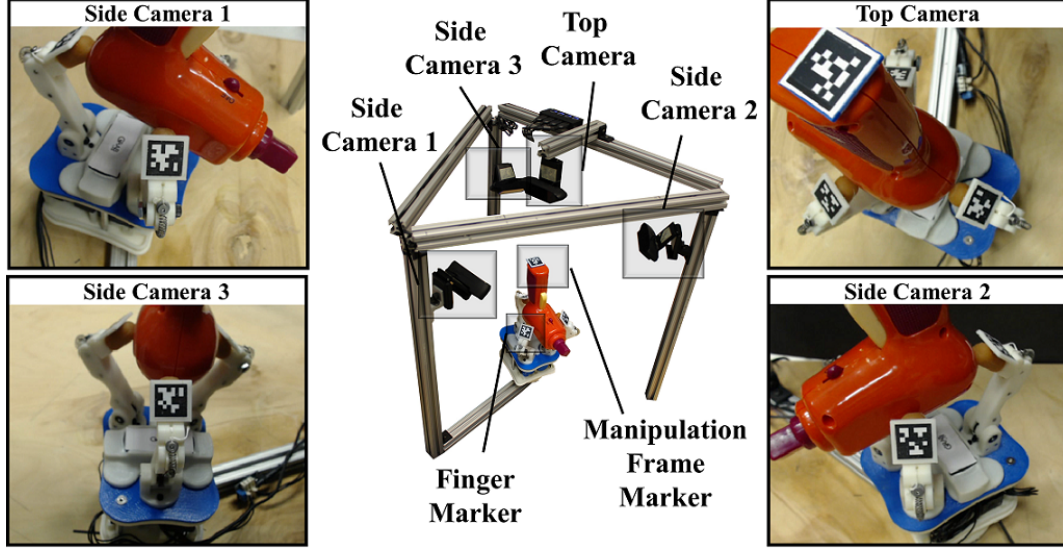


Figure 3.8: A 4-camera tracking system records both, the pose of the grasp frame and the pose of the manipulation frame via attached markers.

transformations from the grasp frame to the manipulation frame as presented in Table 3.3.

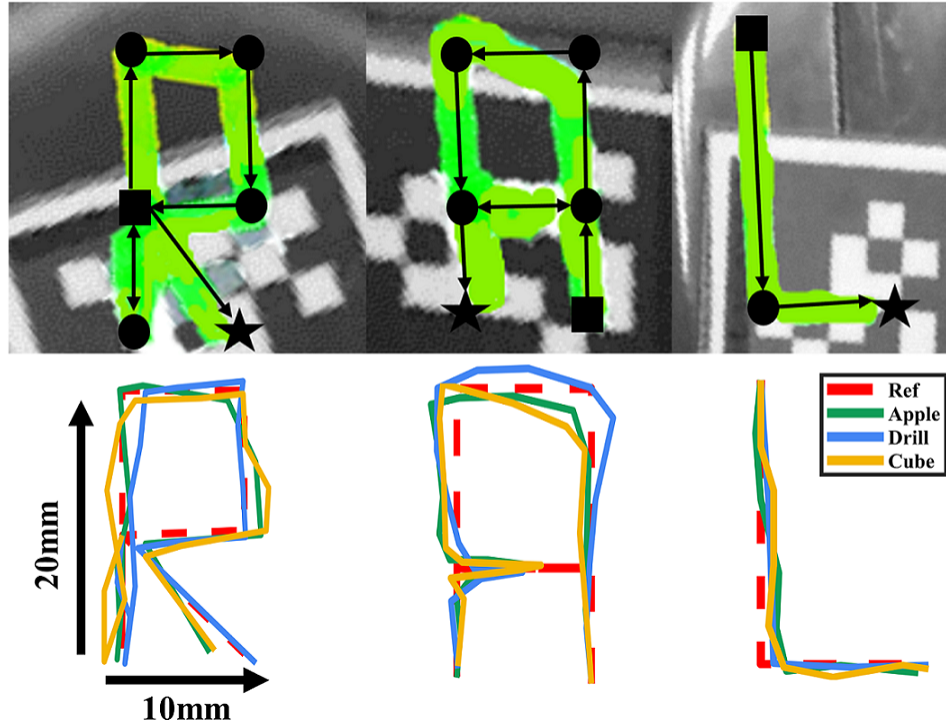
Table 3.3: EXPERIMENT PARAMETERS

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)

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.

We employed a prediction horizon of 3 and set *iter* to 50. As presented in Fig. 3.9, each letter was comprised of a set of goal points, which constructed a system of trajectories approximately 20mm in height and 10mm 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 2mm 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.



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

Figure 3.9: 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.

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 the cross-bar tracing that stopped prematurely. Since we did not greatly



penalize the input actuation velocity, i.e.,  $\lambda$  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'.

### 3.6 Discussions

In this chapter, we addressed the problem controlling partially constrained trajectories for fingertip manipulation 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 to even the difficult task of finger gaiting.

## Chapter 4

# Planning Finger Gaiting for Compliant Hands

### 4.1 Introduction

Within-Hand Manipulation can be characterized as the ability to reorient or reposition an object with respect to the hand frame [1]. The quest for this capability has been studied in the robot manipulation community for decades—from planning and control with rigid hands [17] to efforts with soft, compliant, or underactuated hands [35]. Notably, the majority of these works constrain contacts to remain fixed or rolling during manipulation. This constraint limits the object’s workspace as the actuators can only operate on a single hand-object configuration manifold. Alternatively, finger gaiting, i.e., the process of repositioning contacts on the object during manipulation, can help alleviate such limitations and extend the object’s available workspace.

Finger gaiting is an inherently difficult task for a robot. Given a robot hand, the individual serial link fingers must work in proper unison without collision; maintaining stability while making and breaking contact with the object [72]. The computational complexity of this problem has traditionally been very expensive—requiring the system to calculate and modulate forces and planned joint trajectories online during manipulation. We, conversely, are able to alleviate many of these complexities by leveraging compliance, i.e., safe modal

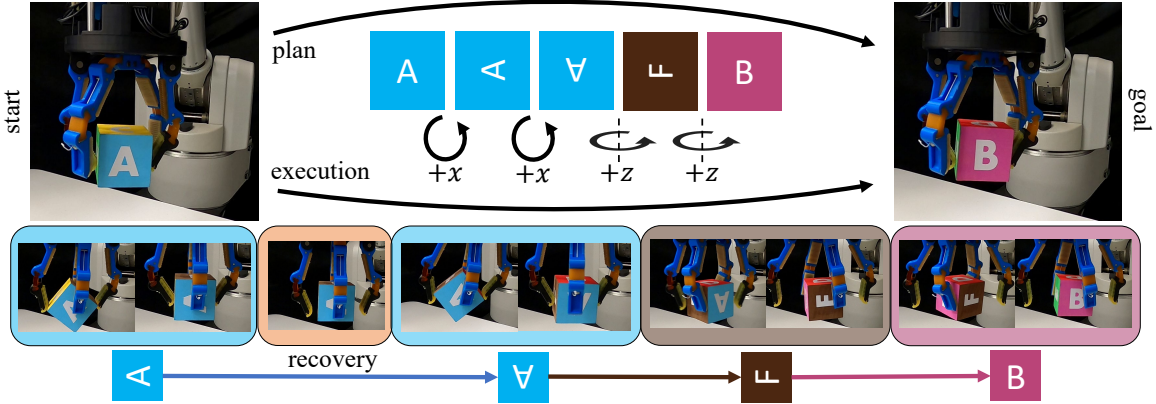


Figure 4.1: We explore the development of a complete  $SO(3)$  planner for within-hand manipulation using finger gaits, by controlling two orthogonal extrinsic rotation axes. Given a start configuration (cube face A), the proposed planner finds an action sequence along the two controlled dimensions so as to reach the desired goal configuration (cube face B). During manipulation, the pose of the object is tracked via a low-latency, 6D pose object tracker, providing feedback for online replanning and disturbance compensation via a recovery phase that uses translation control.

transitions, in our system. The capabilities we present extend beyond what has been illustrated previously in the literature, some purely in simulation [73, 74] and some demonstrated on a real robot, but with support surfaces [11, 75, 76].

In this chapter, we build off the observation that by leveraging the passive adaptive properties of an underactuated hand, we can convert traditional position, force, and stability control problems to a unified motion-only control paradigm, based on the compliant properties of the hand. Concretely, finger gaiting requires contacts to be constantly in motion. The presence and thus activation of certain contacts constrains what forces the hand can impart onto the object. This phenomena creates the abstraction of modes, which can be conceptualized as different families of motion manifolds that are subject to the system’s current set of constraints—typically in the form of gaining or losing contacts. Multi-modal planning within and between these constraints thus becomes a major focus of this work.

We in this work can constrain our planning approach according to the nature of the task. Formally, any orientation in  $SO(3)$  can be achieved via a three action trajectory comprised of two orthogonal rotations. From this formulation, we design two core modal actions for the hand and a multi-modal planner that runs online and is constantly updated via a vision-

based, low-latency 6D pose object tracker [77, 78]. This method continually calculates a trajectory from start to goal and when undesired scenarios arise, such as contact slip, the planner updates and suggests new actions accordingly. We incorporate this approach into an open-source and underactuated hand with four fingers [79]. In the end, we showcase the efficacy of our system through various experiments: tracking the planned and executed trajectory of an object, evaluating the recovery potential given undesired perturbations, and finally, the ability to extend to novel object geometries.

The contributions in this chapter are threefold. First, we develop a *complete* and fast planning solution for in-hand reorientation using two extrinsic rotation axes. Secondly, we describe the utility of compliance for switching between modes and how it “inflates” the contact switching region. And, finally, we present a simple yet effective robot system capable of complex finger gaiting capabilities, underscoring continued discussion in the community on the utility of compliance for in-hand manipulation tasks. In the continuation of this chapter, please refer to Table 4.1 for nomenclature.

Table 4.1: CHAPTER 4 NOMENCLATURE

Symbol	Description
$q$	Particular hand configuration: $q \in \mathbb{R}^6$
$o$	Particular object configuration or pose
$\mathcal{Q}$	Configuration space of the robot: $q \in \mathbb{R}^N$ where $N$ is the number of joints
$\mathcal{O}$	Configuration space of the object: $SE(3)$
$R$	Rotation of the object: $SO(3)$
$\mathcal{M}$	Constrained manifold of the system subject to constraint function $F^\eta(q, o) : \mathbb{R}^N \rightarrow \mathbb{R}^{k_\eta}$

## 4.2 Related Work

### Modeling Manipulation

Modeling robot manipulation has historically been arduous as the dynamics associated with contact are difficult to predict in novel scenarios. From an in-hand manipulation perspective, various levels of modeling have been investigated – from contact models [55] and fingerpad curvature models [47], to hand kinematic models and whole hand-object system models [21]. While these approaches have elucidated many powerful hand-object relationships, inaccurate parameterizations can often lead to task failure [80]. To help alleviate such uncertainty, several end effectors leverage designs that are soft or underactuated [57]. This property provides passive reconfigurability to the mechanism, which can absorb much of the “slack” traditionally required to be fully accounted for in system modeling and can further reduce grasp planning times [81, 82]. In this work, we leverage such mechanisms to unify our planning approach—creating the notion of safe regions for contact switching.

### Extending In-hand Manipulation Capabilities

Models for in-hand manipulation are typically constrained to fixed or rolling contact scenarios. While this advantageously simplifies assumptions for control, it also limits the object’s available workspace to be dependent on the kinematic topology of the hand. Without relying on external contact, e.g, [83], or task-specific hands with roller-based fingers, e.g [84], two general approaches can help alleviate such constraint: sliding manipulation and finger gaiting. Leveraging the former is very difficult, as detecting and controlling its nonlinear conditions requires various levels of advanced sensing [18, 40]. The latter, alternatively, has been largely difficult due to computational considerations, but has been made more successful in recent years [7, 74].

The work in [43] used finger gaits and tactile sensors to maintain grasp stability. [11] used a 24-DOF hand with a motion capture system, in addition to “over 100 years” of simulated data to perform impressive and fast cube manipulation in the palm. [75] leveraged the capabilities of a soft hand with 16 degrees of actuation for similar types of manipulation. While as impressive as these aforementioned works are, we are interested in extending

beyond these capabilities to work against gravity, i.e., maintaining object stability without a support surface, and while utilizing a simple hand with very little onboard sensing.

## Multi-Modal Planning for Manipulation

Problems in robot manipulation are frequently multi-modal, i.e., the seemingly continuous problems have an underlying discrete structure guided by constraints [85]. These constraints are typically imposed by the nature of contact, and by discretizing planning in terms of these manifolds, the planner search space is confined according to physical constraints [86]. Numerous sampling-based multi-modal planners have been described in the literature, which are able to generalize well, particularly in high-DOF scenarios. Though, sampling-based methods without informed exploration are vastly inefficient and suffer from time-complexities associated with over-exploration. Recent work has attempted to address this issue by using informed “leads” to guide exploration [87]. In our work, we build off these observations and constrain our planner’s search according to physical properties of the  $SO(3)$  rotation group. That is, our developed planner constrains the number and nature of mode switching to reduce planning time for continual updates during online execution.

### 4.3 Preliminaries

In this section, we introduce the preliminaries associated with multi-modal motion planning. We first discuss constrained manifolds and then develop notation and terminology for modal switching leveraging compliance.

#### 4.3.1 Constrained Within-Mode Planning

Consider a robot with configuration space,  $\mathcal{Q} \subset \mathbb{R}^N$ , where  $N$  is the number of joints and where  $\mathcal{Q}$  can completely define the state of the robot. Now, consider an object with configuration space,  $\mathcal{O} \subset SE(3)$ , which is located on a support surface, e.g., table top. For simplicity, let’s disregard other potential collisions in the environment. In free space, the current robot configuration,  $q \in \mathcal{Q}$ , is unconstrained and thus able to freely move. Though, when  $q$  causes links of the robot to come in contact with the object in its current

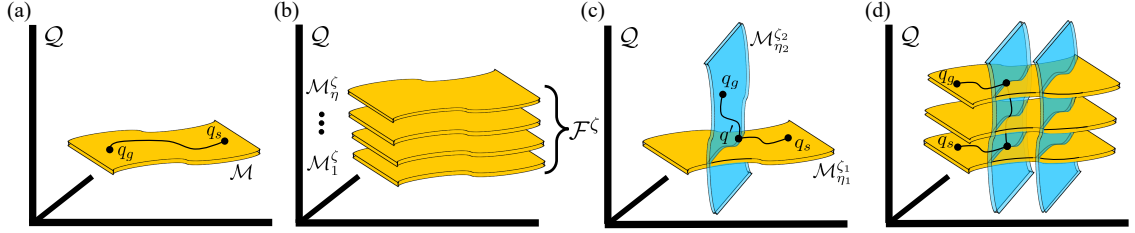


Figure 4.2: Multi-modal planning problems can be conceptualized as operating in different configuration manifolds. (a) Given a single manifold  $\mathcal{M}$ , the planner must find a path along the constrained layer. (b) A single mode, or foliation, can have multiple manipulation manifolds depending on the start configuration of the hand-object system. (c) Switching modes is possible when the system finds a configuration,  $q'$ , that lies on both manifolds. (d) Finding a path from start configuration,  $q_s$ , to goal configuration,  $q_g$ , can require multiple jumps between modes. Note that although represented in this figure as folds, we assume modes to be differentiable but not necessarily euclidean.

configuration,  $o \in \mathcal{O}$ , a constraint is imposed on the system. The robot cannot push the object through the support surface. If, for instance, the goal of the robot is to slide the object along its support surface, the robot's motion is thus constrained to a contact configuration manifold,  $\mathcal{M}$  (Fig. 4.2(a)).

Assume there are several manifolds, indexed by  $\eta$ . The available motion manifold of the robot,  $\mathcal{M}_\eta$ , is subject to its associated constraint function  $F^\eta(q, o) : \mathbb{R}^N \rightarrow \mathbb{R}^{k_\eta}$  where ( $1 \leq k_\eta < N$ ). Modes are thus defined as a family, or foliation,  $\mathcal{F}^\zeta$ , of constrained manifolds that share the same, or similar, constraint functions (Fig. 4.2(b)). Inside each  $\mathcal{F}^\zeta$ , we assume all  $\mathcal{M}_\eta$  to be smooth and differentiable. Therefore, planning in the  $\mathcal{M}_\eta \in \mathbb{R}^{(N-k_\eta)}$  manifold of  $\mathbb{R}^N$  is subject to,

$$\mathcal{M}_\eta = \{q \in \mathcal{Q} | F^\eta(q, o) = \mathbf{0}\} \quad (4.1)$$

### 4.3.2 Safe Mode Transitions

Multi-modal planning requires a robot to transition between at least two constraint manifolds, which likely lie in different foliations, in order to reach the desired goal. Consider a robot starting in configuration  $q_s$ , located in manifold  $\mathcal{M}_{\eta_1}^{\zeta_1}$ . Let's now assume the goal configuration  $q_g$  is located in a different mode and thus along a different manifold, i.e.,  $\mathcal{M}_{\eta_2}^{\zeta_2}$

(Fig. 4.2(c)). In order for a robot to transition to  $q_g$ , a planner must find a candidate submanifold,  $\mathcal{M}_{\eta_1 \cap \eta_2}$ , that enables a transition. Inside of this submanifold lies candidate configuration,  $q'$ , for transitioning. We can define  $\mathcal{M}_{\eta_1 \cap \eta_2}$  according to,

$$\mathcal{M}_{\eta_1 \cap \eta_2} = \{q \in \mathcal{Q} | F^{\eta_1}(q, o) = \mathbf{0}, F^{\eta_2}(q, o) = \mathbf{0}\} \quad (4.2)$$

Note that it is possible for  $\mathcal{M}_{\eta_1 \cap \eta_2}$  to be empty when there exists no  $q'$  to transition  $\mathcal{M}_{\eta_1}^{\zeta_1}$  to  $\mathcal{M}_{\eta_2}^{\zeta_2}$ .

In many robot scenarios, if a transition is possible from  $\mathcal{M}_{\eta_1}^{\zeta_1}$  to  $\mathcal{M}_{\eta_2}^{\zeta_2}$ , the transfer region typically has a volume near zero, i.e.,  $\Omega(\mathcal{M}_{\eta_1 \cap \eta_2}) \approx 0$  where  $\Omega$  is the convex hull function. There are situations, however, where we can relax this constraint to form *safe mode transitions*. Formally, the manifold's constraint function,  $F^\eta(q, o)$ , is dependent on the robot's current configuration,  $q$  and object configuration,  $o$ . Though due to near-unmodelable DOFs in soft, compliant, or underactuated robots or objects, the controlled configuration of the robot may still hold for  $\mathbb{R}^N \rightarrow \mathbb{R}^{k_\eta}$ , but the true configuration of the system may be non-deterministic. Thus, there can exist a relaxation of this traditional modal switching constraint, which in turn can simplify and accelerate planning [81]. We define a modal transition threshold vector,  $\rho$ , that defines an inflation of the constraint manifold along different axes, which inherently increases the potential volume of  $\mathcal{M}_{\eta_1 \cap \eta_2}$  to develop a *safe transfer region*,

$$\mathcal{M}_{\eta \cap \eta'}^* = \{q \in \mathcal{Q} | -\rho < F^\eta(q) < \rho, -\rho < F^{\eta'}(q) < \rho\} \quad (4.3)$$

The success of our finger gating platform in this work largely leverages this concept. Explicitly defining  $\rho$ , however, is difficult as it is system dependent, i.e., properties of the hand and the object determine how inflated the switching regions become. As a proof-of-concept analysis, we estimate  $\rho$  via experimentation in Sec. 4.7 and will leave computational investigation for future work.



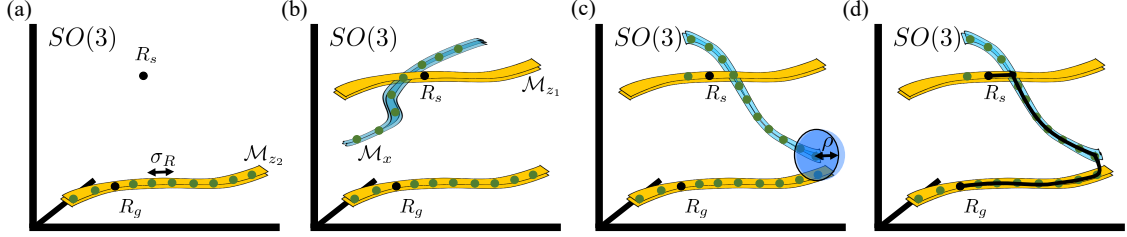


Figure 4.3: (a) Our planning solution for  $SO(3)$  begins by enumerating outward the goal orientation manifold by step size  $\sigma_R$  and creating a KD-Tree. (b) After creation, a forward search outward is performed from the start orientation,  $R_s$ , combining a  $z$ -axis rotation first, then an  $x$ -axis rotation. (c) This search continues until a candidate orientation is within some distance  $\rho$  from the expanded goal manifold. (d) Finally, the entire plan is enumerated from start to goal along the acquired trajectory with appropriate timestamps and step size  $\sigma_R$ .

## 4.4 Orientation-based Motion Planning

We formally define the problem of orientation-based motion planning and the minimum number of actions required. Then, we utilize this as a guarantee for the completeness our planner’s modal search transitions.

### 4.4.1 Proper Rotations in $SO(3)$

*Proper rotations* are sets of angles that can fully define any rotation in the group,  $SO(3)$ . Let’s assume we have an object with some rotation,  $R$ . It is possible to transition  $R$  to any rotation configuration via a combination of rotations along two orthogonal axes. Though there are six total combinations, we will instantiate our planner to focus on rotations along the  $z$ -axis ( $R_z$ ) and  $x$ -axis ( $R_x$ ) to serve as an example.

---

**Theorem 1:** *Let the rotational configuration of the object be  $R \in SO(3)$ . Then there exists at least one  $(\phi, \theta, \psi) \in [0, 2\pi) \times [0, \pi] \times [0, 2\pi)$  such that  $R = R_z(\phi)R_x(\theta)R_z(\psi)$  [88, 89].*

This realization defines the minimum number of unique modes that are needed to achieve any rotation in  $SO(3)$ . In this work, we will generally focus on transitioning between two modes,  $\mathcal{F}^{R_z}$  and  $\mathcal{F}^{R_x}$ , a total of two times to reach any goal configuration.

**Proof:** Consider the group of rotations  $SO(3) = \{R \in \mathbb{R}^{3 \times 3} | RR^T = I, \det(R) = 1\}$ , where there exists elemental rotations  $R_x \in SO(3)$  and  $R_z \in SO(3)$ , about the  $x$ - and  $z$ - axes,

respectively. We can decompose  $R$  to be a tuple of three unit vectors, i.e.,  $R = [\hat{v}_1 \hat{v}_2 \hat{v}_3]$ , that represent a right handed triple coordinate frame. Let's assume we provide a basis for  $\mathbb{R}^3$  also representing a proper right handed triple,

$$\hat{e}_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \quad \hat{e}_2 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \quad \hat{e}_3 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \quad (4.4)$$

From this formulation, we can set a reference vector  $\hat{n}$ , which is perpendicular to  $\hat{v}_3$  as,

$$\hat{n} = \begin{cases} \frac{\hat{e}_3 \times \hat{v}_3}{|\hat{e}_3 \times \hat{v}_3|} & \text{if } \hat{v}_1 \neq \pm \hat{e}_3 \\ \text{sgn}(\hat{v}_1) \hat{e}_1 & \text{else} \end{cases} \quad (4.5)$$

where  $\text{sgn}(\cdot)$  is the sign function and  $\times$  is the cross product. Given  $\hat{n}$  and our first basis vector  $\hat{e}_1$ , where  $\hat{n}$  is also perpendicular to  $\hat{v}_3$ , there exists some rotation angle  $\phi \in [0, 2\pi)$  along the  $z$ -axis which maps  $\hat{n}$  to  $\hat{e}_1$ , but also leaves  $\hat{e}_3$  invariant according to Euler's theorem, i.e.,

$$\hat{n} = R_z(\phi) \hat{e}_1 \quad \hat{e}_3 = R_z(\phi) \hat{e}_3 \quad (4.6)$$

After application of  $\phi$ ,  $\hat{e}_3$  and  $\hat{v}_3$  are now both perpendicular unit vectors to  $\hat{n}$ . Thus, there exists a new offset angle  $\theta$  along the  $x$ -axis that will align these two vectors. Notably,  $\theta \in [0, \pi]$  because  $\hat{n}$  began with direction  $\hat{e}_3 \times \hat{v}_3$  and thus has a antipodal mirrored coordinate on the great circle. Consequently, the new formulation is,

$$\hat{n} = R_z(\phi) R_x(\theta) \hat{e}_1 \quad \hat{v}_3 = R_z(\phi) R_x(\theta) \hat{e}_3 \quad (4.7)$$

Finally, we note that  $\hat{n}$  and  $\hat{v}_1$  are now both orthogonal to basis vector  $\hat{e}_3$ . Therefore, there must exist an angle  $\psi \in [0, 2\pi)$  between these two vectors such that there becomes alignment. A rotation of  $\psi$  about  $\hat{v}_3$  is thus implemented,

$$\hat{v}_1 = R_z(\phi) R_x(\theta) R_z(\psi) \hat{e}_1 \quad \hat{v}_3 = R_z(\phi) R_x(\theta) R_z(\psi) \hat{e}_3 \quad (4.8)$$

Notably, this above formulation does not alter or constrain the original unit vector  $\hat{v}_2$  in any way. We find that  $R_z(\phi)R_x(\theta)R_z(\psi)$  has the same first and third columns as our target rotation,  $R$ . Due to the group properties of  $SO(3)$ ,  $R_z(\phi)R_x(\theta)R_z(\psi)$  and  $R$  must have the same second column, i.e.,  $R_z(\phi)R_x(\theta)R_z(\psi)\hat{e}_2$ .

■

---

#### 4.4.2 Planning Orientation Transitions

Following this realization and the preliminaries in Sec. 4.3, we develop a *complete* multi-modal planning algorithm for finger gaiting that consists of three rotations along two orthogonal axes. If, for instance, all three orthogonal axes were available, the planning problem would be mostly trivial—simply transition the object along each axis as much as necessary to reach the goal. For the duration of this paper, we will refer to rotations about axes as being extrinsic, i.e., according to the hand frame of robot, and will adopt the notation that  $R_x(\cdot)$  and  $R_z(\cdot)$  are rotations of the object about the  $x$ - and  $z$ -axes of the hand frame, or roll and yaw, respectively.

Let’s assume the capabilities of the robot enable the formulation of just two mode foliations, which are able to take an object in *any* configuration and rotate about the  $x$ - and  $z$ -axes, respectively. Given an object in start configuration  $o_s = \{T_s \in \mathbb{R}^3, R_s \in SO(3)\}$ , the objective of this planner is to reach a desired goal pose,  $o_g = \{T_g \in \mathbb{R}^3, R_g \in SO(3)\}$ . For now, we will disregard the translational positions,  $T_s$  and  $T_g$ , and focus on rotation.

As to accelerate computation, the proposed planning method is *bi-directional* in that it builds a search trajectory outward from both, the start and goal configurations. More formally, there exists rotations such that  $R_g = R_{z_2}(\phi)R_x(\theta)R_{z_1}(\psi)R_s$  where  $R_{z_1}(\psi)$  is the first rotation about the  $z$ -axis,  $R_x(\theta)$  is the rotation about the  $x$ -axis, and  $R_{z_2}(\phi)$  is the second rotation about the  $z$ -axis. Thus, the goal of this planner is to solve for  $\psi, \theta$ , and  $\phi$  and enumerate a trajectory accordingly.

Alg. 4 begins with the backward pass by populating a  $z$ -axis rotation outward from  $R_g$

---

**Algorithm 4**  $SO3Plan(\cdot)$ 

---

**Input:**  $R_s, R_g, \rho, \sigma_R$   $\triangleright$  object start orientation, object goal orientation, modal transition threshold, rotational step size

**Output:**  $\mathcal{P}$   $\triangleright$  multi-modal plan

- 1:  $\Sigma = [0, \sigma_R, -\sigma_R, 2\sigma_R, -2\sigma_R, \dots, \pi, -\pi]$   $\triangleright$  candidate steps
- 2:  $\mathcal{M}_{z_2} \leftarrow [R_z(\phi)R_g \text{ for } \phi \text{ in } \Sigma]$   $\triangleright$  expanded goal manifold
- 3:  $KD \leftarrow KDTree.build(\mathcal{M}_{z_2})$   $\triangleright$  KD-Tree of goal manifold
- 4: **for**  $\psi$  **in**  $\Sigma$  **do**
- 5:     **for**  $\theta$  **in**  $\Sigma$  **do**
- 6:          $R^* = R_x(\theta)R_z(\psi)R_s$   $\triangleright$  outward expanding search
- 7:          $\phi \leftarrow KD.query\_closest\_config(R^*)$
- 8:         **if**  $dist_R(R_z(\phi)R_g, R^*) < \rho$  **then**  $\triangleright$  distance function
- 9:             **break loops**  $\triangleright$  bidirectional trajectory connects
- 10:         **end if**
- 11:     **end for**
- 12: **end for**
- 13:  $\mathcal{P} \leftarrow \text{planner.enumerate}(R_s, \sigma_R, \psi, \theta, \phi)$   $\triangleright$  enumerate path outward according to connection (Fig. 4.3(d))
- 14: **return**  $\mathcal{P}$

---

with step size,  $\sigma_R$ , which is user-defined (Fig. 4.3(a)). For our implementation,  $\sigma_R$  can generally range anywhere from 0.5-5.0° and can be adaptive according to system architectures. Let's say that these discretized rotations outward from  $R_g$  can sufficiently define the entire manifold,  $\mathcal{M}_{z_2}$ , i.e., an entire rotation of the  $z$ -axis about  $R_g$ . From these configurations in  $\mathcal{M}_{z_2}$ , we can build a KD-Tree,  $KD$ , which is a data structure that enables efficient querying of the rotations included in this manifold.

Now that the backwards direction is instantiated, the goal of the forward pass is to find rotations  $R_x(\theta)R_{z_1}(\psi)R_s$  such that they can “connect” to the manifold  $\mathcal{M}_{z_2}$  represented by  $KD$  within some distance threshold  $\rho$  from (4.3) (Fig. 4.3(c)). As aforementioned,  $\rho$  is the inflation factor for contact switching, and is generally system dependent. Here, we utilize a user-defined rotational distance function  $dist_R(\cdot)$ , where for our instantiation, we calculate the L<sup>2</sup>-norm of the difference vector along the roll, pitch, and yaw dimensions of two rotations. Though, we note that there are other metrics available, e.g., [90]. Tangibly, this forward pass sequentially searches through candidate values for  $\psi$  and  $\theta$  until the trajectory reaches the goal manifold  $\mathcal{M}_{z_2}$ , which is found by continually querying  $KD$ . Once this connection is found (Fig. 4.3(d)), values for  $\psi$ ,  $\theta$ , and  $\phi$  are recorded, and a

trajectory for plan  $\mathcal{P}$  is enumerated with timestamps along step size  $\sigma_R$  from start to goal.

## 4.5 Object Control

Finger-gaiting is a dynamic and uncertain process; making and breaking contacts can often lead to slip which could further result in object ejection. To mitigate such occurrences, our solution is to continually replan and execute modal actions. More specifically, we rely on an online servoing-based framework so as to reach the desired goal. When the object enters a region of potential instability, such as being distant from the workspace center, a recovery phase comprised of translational modal actions is enacted to relocate the object.

### 4.5.1 Translational Planning

Planning translations within-hand, presented in Alg. 5, performs a simple, greedy visual servoing control approach. Generally, the algorithm calculates the translational difference,  $\gamma$ , along each of the coordinate axes from start position,  $T_s$ , and goal position,  $T_g$ . The planner then chooses the direction of largest difference and creates a plan to take a single step in that direction [50].

---

#### Algorithm 5 *TranslationPlan*( $\cdot$ )

---

**Input:**  $T_s, T_g, \sigma_T$  ▷ object position, object goal position, step size  
**Output:**  $\mathcal{P}$  ▷ plan

- 1:  $\gamma.x \leftarrow T_g.x - T_s.x$  ▷ translational difference along  $x$ -axis
- 2:  $\gamma.y \leftarrow T_g.y - T_s.y$  ▷ translational difference along  $y$ -axis
- 3:  $\gamma.z \leftarrow T_g.z - T_s.z$  ▷ translational difference along  $z$ -axis
- 4:  $\mathcal{P} \leftarrow \{\}$
- 5:  $\pi \leftarrow \max(|\gamma.x|, |\gamma.y|, |\gamma.z|)$  ▷ direction of maximum deviation
- 6: **if**  $\pi$  **is**  $|\gamma.x|$  **then**
- 7:  $\mathcal{P} \leftarrow [T_x + \text{sgn}(\gamma.x)\sigma_T, T_y, T_z]$  ▷ desired move  $\pm T_x$
- 8: **else if**  $\pi$  **is**  $|\gamma.y|$  **then**
- 9:  $\mathcal{P} \leftarrow [T_x, T_y + \text{sgn}(\gamma.y)\sigma_T, T_z]$  ▷ desired move  $\pm T_y$
- 10: **else if**  $\pi$  **is**  $|\gamma.z|$  **then**
- 11:  $\mathcal{P} \leftarrow [T_x, T_y, T_z + \text{sgn}(\gamma.z)\sigma_T]$  ▷ desired move  $\pm T_z$
- 12: **return**  $\mathcal{P}$

---

### 4.5.2 Orientation and Translation Control

Our control approach combines both, Alg. 4 and Alg. 5. This method requires a goal threshold,  $\tau$ , a linear interpolant update parameter,  $\lambda$ , and *a priori* knowledge of the hand’s workspace center,  $T_{center}$ , for object recovery.

In executing Alg. 6, orientation control is accomplished first. A plan  $\mathcal{P}$  is found according to the object’s current object rotation,  $R$ . The robot then takes a *single* action along  $\mathcal{P}$  via the `robot.mode_action( $\cdot$ )` method. This system-specific function enacts a motor position step proportional to  $\sigma_R$  along the desired mode. From this action, an object pose displacement,  $\delta$ , is calculated by taking the distance between the original orientation,  $R$ , and the new orientation,  $R'$ , after the action. This value  $\delta$  serves as a differential update term along the orientation transition, which allows us to adaptively re-estimate  $\sigma_R$  according to an interpolant update parameter  $\lambda$ . Since, as aforementioned, there is inherent uncertainty in the task of finger gaiting, this adaptive update to the transition step size  $\sigma_R$  serves to aid in our plan’s accuracy for the true motion of the hand-object system. The algorithm continues while  $R$  is outside of the goal distance threshold  $\tau_R$ . Notably, a recovery phase (lines 12-16) is developed inside of this loop and is enacted when the object position,  $T$ , is distance  $\tau_T$  outside of  $T_{center}$ . At this point, the system takes greedy transition steps towards the center of the workspace for recovery.

Alg. 6 concludes by performing steps along  $\mathcal{P}$  found from the `TranslationPlan( $\cdot$ )` method so as to reposition the object within-hand after object reorientation. We implement this method last to decouple orientational and translational planning for the final pose.

### 4.5.3 Generalization and Practical Algorithm Modifications

The authors present this solution as a generalized approach to planning for in-hand manipulation. Surely, the design and realization of the `robot.mode_action( $\cdot$ )` method begs questions for system-specific scenarios, but can be found in a variety of ways: analytical modeling, dynamics learning, reinforcement learning, etc. While we do not focus on describing the underlying components of this function in this work, we largely leverage an energy-modeling technique outlined in [91] for determining predicted object transitions.

---

**Algorithm 6** *ObjectControl*( $\cdot$ )

---

**Input:**  $o_g, \rho, \sigma, \tau, \lambda, T_{center}$ 

▷ object  
goal pose, modal transition threshold vector, pose transition step size, goal pose  
reached threshold, interpolant step size, center of object workspace

**Output:**  $o_f$  ▷ final object pose

- 1:  $T_g, R_g \leftarrow o_g.T, o_g.R$  ▷ translation and rotation of goal pose
- 2:  $o \leftarrow \mathbf{tracker.perceive}()$  ▷ object pose  $\in SE(3)$
- 3:  $T, R \leftarrow o.T, o.R$  ▷ translation and rotation of object pose
- 4: **while**  $dist_R(R_g, R) > \tau_R$  **do** ▷ orientation control
- 5:      $\mathcal{P} \leftarrow \mathbf{planner.SO3Plan}(R, R_g, \rho, \sigma_R)$  ▷ Alg. 4
- 6:      $\mathbf{robot.mode\_action}(\mathcal{P}, \sigma_R)$  ▷ execute modal action
- 7:      $o' \leftarrow \mathbf{tracker.perceive}()$  ▷ new object pose
- 8:      $T', R' \leftarrow o'.T, o'.R$  ▷ new object translation and rotation
- 9:      $\delta \leftarrow dist_R(R', R)$  ▷ size of last action step
- 10:      $\sigma_R \leftarrow \sigma_R + \lambda(\sigma_R - \delta)$  ▷ update rotational transition step
- 11:      $T, R \leftarrow T', R'$
- 12: **while**  $dist_T(T, T_{center}) > \tau_T$  **do** ▷ object recovery
- 13:      $\mathcal{P} \leftarrow \mathbf{planner.TranslationPlan}(T, T_{center}, \sigma_T)$
- 14:      $\mathbf{robot.mode\_action}(\mathcal{P}, \sigma_T)$  ▷ execute modal action
- 15:      $o \leftarrow \mathbf{tracker.perceive}()$  ▷ object pose
- 16:      $T, R \leftarrow o.T, o.R$  ▷ object translation and rotation
- 17: **end while**
- 18: **end while**
- 19: **while**  $dist_T(T_g, T) > \tau_T$  **do** ▷ translation planning
- 20:      $\mathcal{P} \leftarrow \mathbf{planner.TranslationPlan}(T, T_g, \sigma_T)$  ▷ Alg. 5
- 21:      $\mathbf{robot.mode\_action}(\mathcal{P}, \sigma_T)$  ▷ execute modal action
- 22:      $o \leftarrow \mathbf{tracker.perceive}()$  ▷ object pose
- 23:      $T, R \leftarrow o.T, o.R$  ▷ object translation and rotation
- 24:  $o_f \leftarrow \mathbf{tracker.perceive}()$  ▷ final object pose
- 25: **return**  $o_f$

---

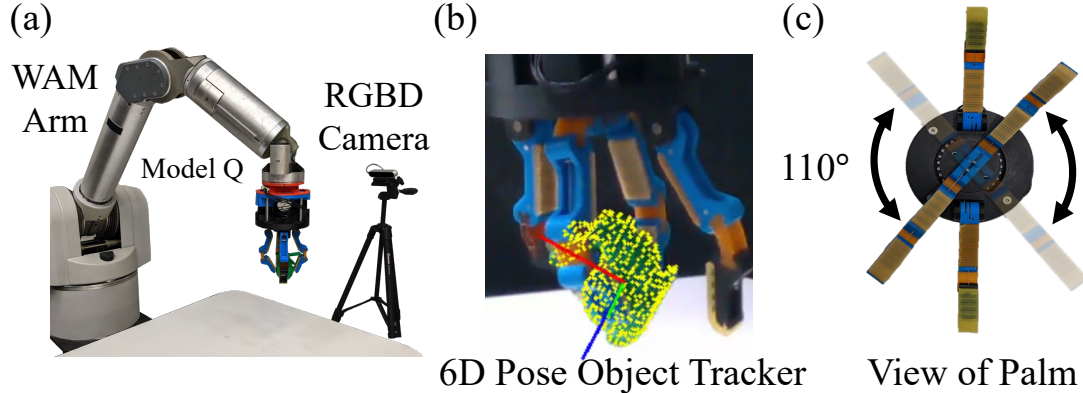


Figure 4.4: (a) Our system is comprised of a RGBD camera fixated to the robot’s environment. (b) During manipulation, the pose of the object, e.g., bunny, is tracked via a 6D pose object tracker. (c) A bottom view of the Yale Model Q illustrates the  $110^\circ$  abduction capabilities of the differentially coupled fingers.

Moreover, the proposed planner can be further accelerated by subtle variations to the aforementioned algorithms. First,  $KD$  only needs to be computed once since the goal pose is static. Also, initialized step sizes for  $\sigma$  can deviate between modes according to system-specific properties, e.g., when gravity affects the rotation for one mode more than another.

## 4.6 Experimental Setup

### 4.6.1 Robot Setup

We utilize the open-source Model Q, an underactuated hand from the Yale OpenHand Project [49]. The hand is equipped with four two-link fingers and four total actuators. A single motor controls the actuation of an opposing set of fingers with joints comprised of flexures, guided by a differential. This enables passive reconfigurability between these two fingers and into/out-of the plane of grasping. These two coupled fingers are connected to an actuated rotary joint located within the palm, allowing  $110^\circ$  rotation about the  $z$ -axis of the hand (Fig. 4.4). The final two motors serve to actuate the remaining two fingers individually via tendons. Being underactuated, the hand’s joint configuration cannot be determined directly, as it is not equipped with joint encoders or tactile sensors. See [79] for additional design information.



The hand is mounted on a 7-DOF Barrett WAM arm. External to the robot, an RGBD Intel Realsense D415 camera is calibrated to the robot's environment. Its imagery is processed through the low-latency (60Hz) 6D pose object tracker (Fig. 4.4). The tracker is robust to finger occlusion, has approximately  $\pm 1.5\text{mm} / 2^\circ$  of noise along any axis, and is trained solely with synthetic object data. See [77] for additional information.

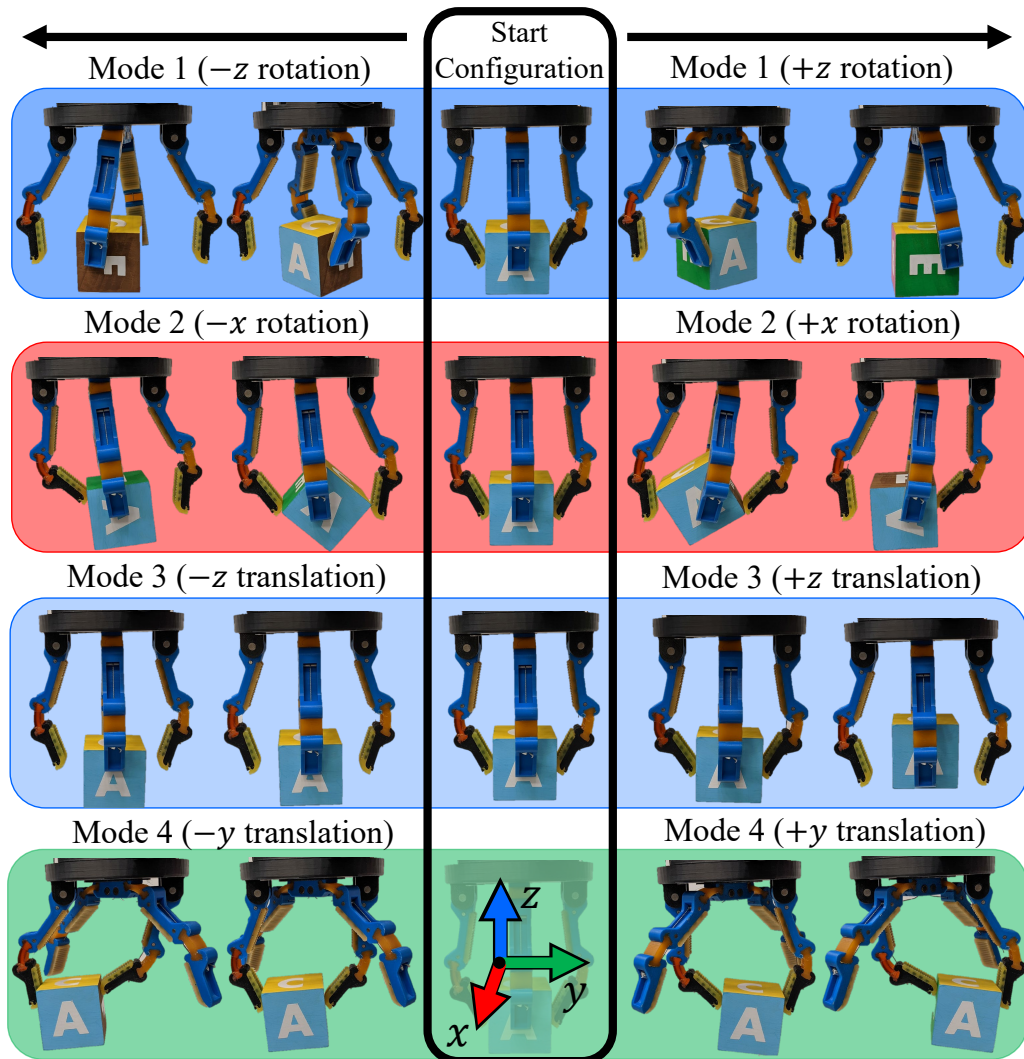


Figure 4.5: The Yale OpenHand Model Q is capable of operating along four modes for fingertip-based manipulation. Modes are shown with transitions from the start configuration (center).



Figure 4.6: (a) We experimentally validate our method with 5 different objects, namely a cube, sphere, toy truck, Stanford Bunny, and toy duck. (b) Tessellated faces of the cube show poses of the affixed letters.

#### 4.6.2 Mode Design

As described in Sec. 4.4.2, a series of rotations about two orthogonal axes allow an object to reach any orientation in  $SO(3)$ . Given the hand topology of the Model Q, an extrinsic  $z$ -axis rotation is easily possible via the palm’s rotary joint. Moreover, an extrinsic  $x$ -axis rotation is also possible via coordinated finger motions; grasping with the differentially actuated opposing pair and pushing the object from either individually driven fingers as validated by [91].

In addition to the two rotational motions, the hand topology furthermore allows for only two translational modes. The first can translate the object along the  $z$ -axis by squeezing the object towards the palm and regrasping, or by leveraging small amounts of coordinated slip, and simple  $y$ -axis translation is possible via a method described in [50] (Fig. 4.5). Note that translation about the  $x$ -axis is not possible due to the actuation coupling of the rotary set of fingers. By combining these two translational modes, we are able to provide a recovery phase for the object (Fig. 4.1, 4.8), repositioning the object towards the middle of the workspace so that manipulation can safely continue. Note that  $x$ -axis translation is thus omitted.

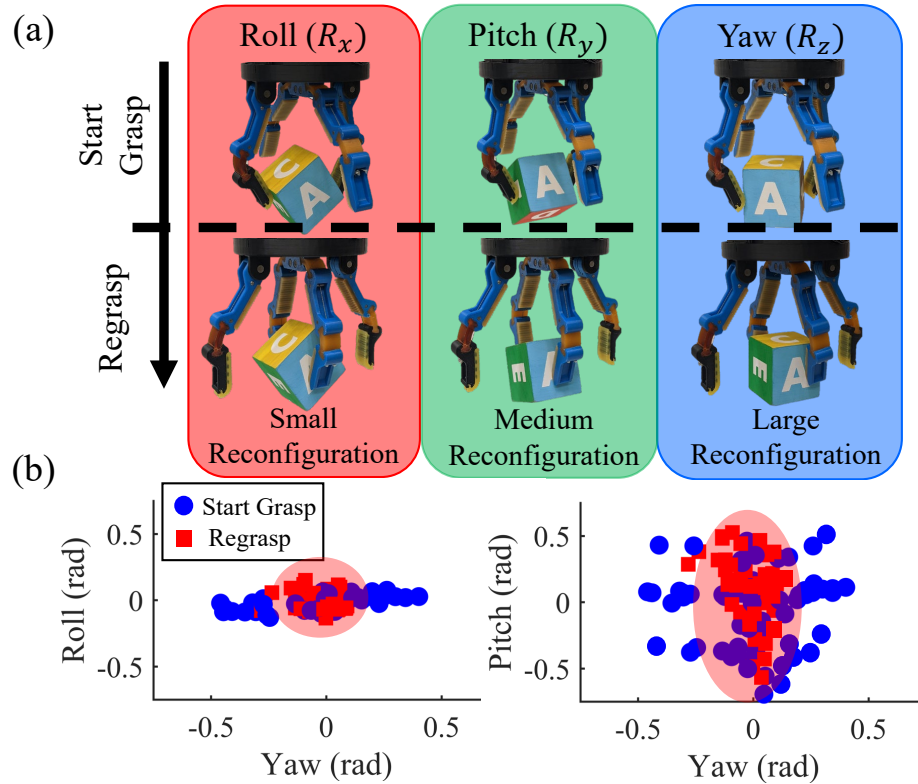


Figure 4.7: The safety of mode transitions are attributed to the reconfigurability of the underactuated mechanism. (a) Starting rotations around different axes of the object elicit different amounts of reconfiguration upon regrasping. (b) This is particularly apparent in the yaw ( $z$ -axis) direction of the object, where the object follows the minimum energy configuration of the mechanism, and hence allowing us to estimate  $\rho$  from Eq. (4.3).

## 4.7 Experiments

### 4.7.1 Safe Modes Characterization

The passive adaptive nature of compliant hands is particularly beneficial for the task of finger gaiting. Simply, small errors associated with state estimation, modeling, and control can likely be dismissed by the system “absorbing” the slack. It moreover determines the “inflation” of switching regions, denoted by  $\rho$ . To help quantify and validate this adaptive nature, we evaluate how a *regrasp* action, i.e., transferring the grasp between sets of opposing fingers, affects the resultant object configuration.

Evaluated with the cube (Fig. 4.6), we perturb the object along each axis. Subsequently, the hand transfers the grasp to the opposite set of opposing fingers. In doing so, we record the amount of object reconfiguration along each axis via the object tracker. Notably, with the roll orientation of the object, we see a small amount of reconfiguration, but in the yaw orientation, we note the most reconfiguration (Fig. 4.7), due to the geometric properties of the cube. More specifically, from the start grasp (blue circle), to the final regrasped configuration (red square), the object tends to fall towards the minimum energy configuration of the system (red region) [7, 91]. We note that although there is a large variability in the object’s start orientation ( $\pm 0.5$  radians), the hand is able to regrasp the object successfully in all cases. This underscores our *safe modes* discussion, and generally allows us to estimate how large  $\rho$  can be. For safety, we define  $\rho = 0.2$  radians.

### 4.7.2 Single Trajectory Execution

We evaluate the repeatability of executing a single trajectory of the cube. Here, the goal of the task is to rotate the cube from face A to face B, and when completed, translate the object 1.8cm along the negative  $y$ -axis. The resultant executions, depicted in Fig. 4.8(a), shows the average trajectory of the object over 8 different trials. During this task, we note how closely trajectories follow along the roll and yaw dimensions, as these are controlled via our safe modes.

Object recovery occurs at similar times in the trials, where the system transitions the object along the positive  $z$ -axis before completing the  $x$ -axis rotations (Fig. 4.8(b)). At

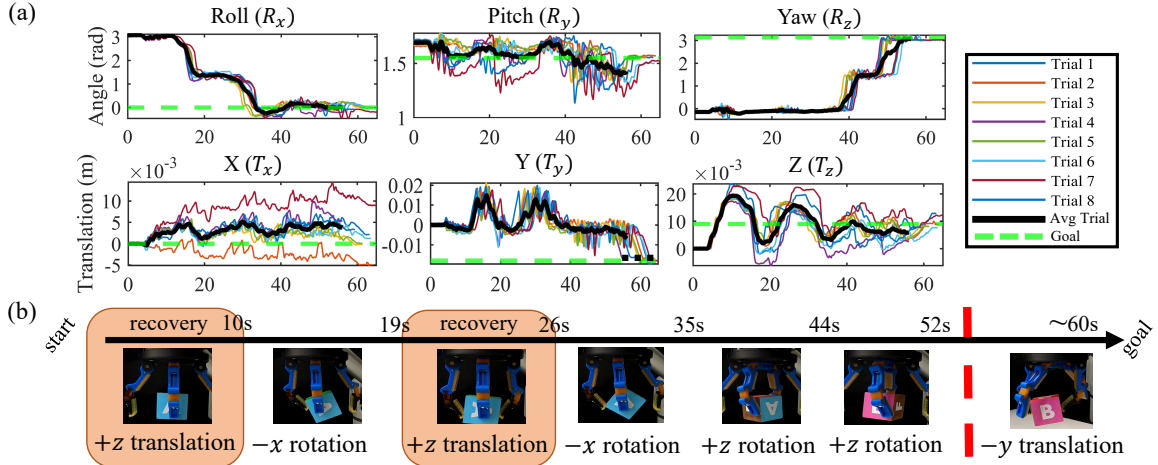


Figure 4.8: We execute a single planned trajectory 8 times and record its repeatability. (a) Controlled dimensions of the object trajectory, such as roll and yaw, follow closely in all trials, where as uncontrolled dimensions, such as  $x$ -axis translation and pitch, are allowed to drift. (b) Modes are enacted at different times in the manipulation, including recovery phases when the object is not in the center of the graspable workspace.

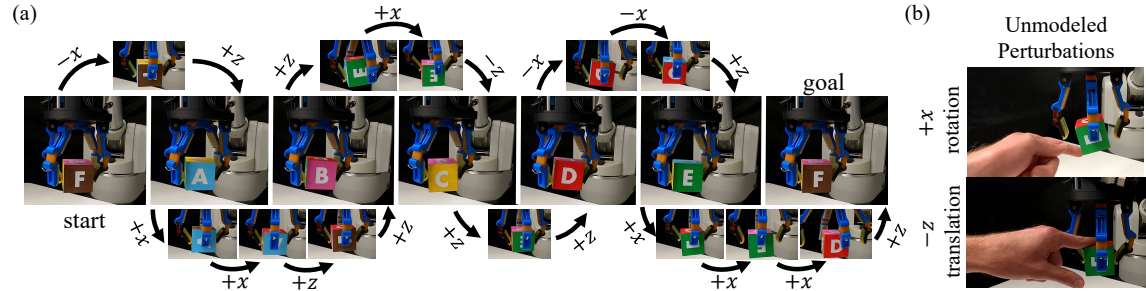


Figure 4.9: We test system robustness by (a) following through an extended trajectory of cube faces F, A, B, C, D, E, and F once again, and (b) deliberately applying different perturbations to the object along controlled dimensions so as to require online replanning and recovery.

the end of the orientation control sequence, the object is appropriately translated along the  $y$ -axis to reach its desired goal pose. The average trajectory data, depicted in solid black, does not extend past the 55 second mark due to one trial finishing earlier than others. The authors have, for clarity, extended the trajectory in the  $y$ -axis translation using a dotted line to illustrate the average final pose.

### 4.7.3 Continuous Goal Trajectories and Robustness

We showcase the robustness of our method by evaluating both, an extended control trajectory and by disturbing the object pose during execution. As depicted in Fig. 4.9(a), we

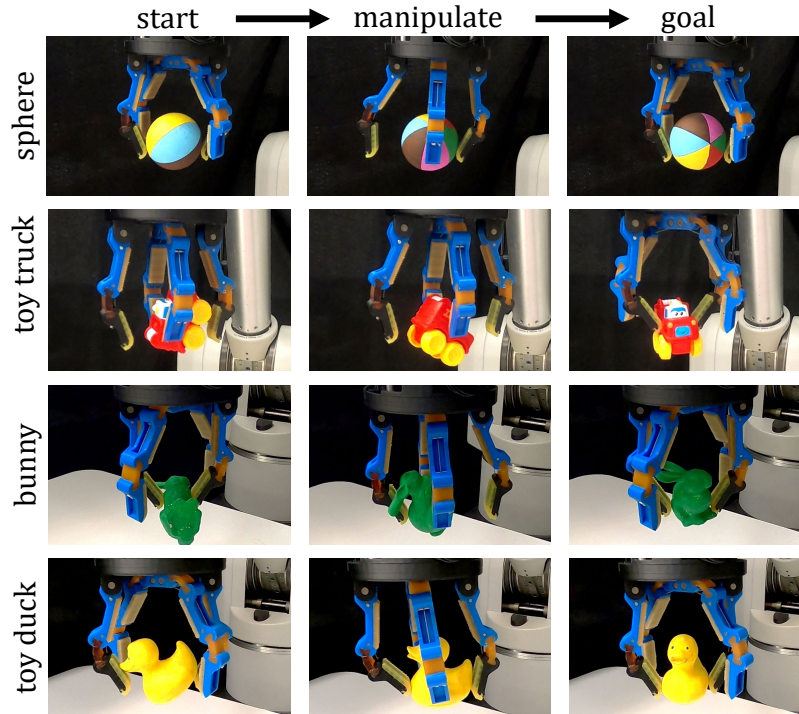


Figure 4.10: Our manipulation planner is able to extend to objects of convex and non-convex geometries, and reach any orientation in  $SO(3)$ .

execute a trajectory starting on cube face F, transitioning through faces A,B,C,D,E and finally reaching the face F once again. In all cases, the object was able to reach the goal face within  $8^\circ$ .

Moreover, we illustrate our ability to replan control trajectories when unmodeled perturbations occur (Fig. 4.9(b)). At different points during manipulation, we randomly perturb the object pose 13 total times along the least constrained dimensions, the  $x$ -axis rotation and the  $z$ -axis translation. Recovering and adapting to these occurrences over the course of a 205 second trajectory execution, the system completes the object reorientation task within  $6^\circ$  of the goal face.

#### 4.7.4 Generalization to Object Geometry

Finally, we test the transferability of our method by varying object geometries (Fig. 4.6). In doing so, recovery properties, such as  $T_{center}$  from Alg. 6, were recalculated, and the 6D pose object tracker was retrained with new synthetic object data; other aforementioned parameters, including  $\rho$ , remained the same. Depicted in Fig. 4.10, we evaluate simple,

Table 4.2: EXPERIMENT EVALUATION

Obj.	$err$ (deg)	Plan time (s)	Total time (s)	Success
sphere	4.6	13.2	153.1	5/5
truck	4.2	10.8	112.3	5/5
bunny	9.5	14.0	128.0	2/5
duck	8.9	7.1	73.2	3/5

Experimental evaluation of four different objects with error in final orientation, planning time, total time, and number of execution successes.

purely rotational trajectories about each of the objects. Notably, the sphere easily followed each of its guided reference trajectories, and was able to reach goal orientations within  $\pm 5^\circ$  along any axis. The toy truck, further illustrated our capabilities, transitioning to its goal configuration in approximately 110 seconds with 4 recovery phases. And finally, the Stanford Bunny and the toy duck were the most difficult to manipulate due to their non-convex properties. With the bunny, fingers would often get stuck behind its ears while making and breaking contact. Consequently, this occurrence caused the hand-object system to get stuck in specific object orientations and required restart. Moreover, during duck manipulation, the size of the duck’s head affected the object’s center of mass. This made some actions, such as  $x$ -axis rotation, difficult as the motion was then more dynamic in nature. Only a simple, two-rotation trajectory could be completed with the duck, resulting in a shorter execution time. The precision of these two manipulations was decreased to  $\pm 10^\circ$  along any rotation axis and were consequently not quite as robust as the other objects. We provide benchmarking baselines for our experimentation in Table 4.2 according to guidelines outlined [92]. Please refer to the supplementary video for experimental evaluations.

## 4.8 Discussions

In this chapter, we developed a robust and *complete* solution to  $SO(3)$  finger gait planning. While instantiated on an underactuated hand in this work, the methods described can generalize to other hand-object systems. We showcased our method with numerous tasks—highlighting the safety of our modes, the repeatability of our trajectories, the robustness of our planning and control approach, and our ability to generalize to new object scenarios.

This work builds on a promising approach of vision-based control for compliant within-hand manipulation. The tasks completed in this letter extend beyond what has been possible in previous works.

We would like to highlight the larger contribution of this work in supporting the thesis that compliance helps speed up planning for robot manipulation. Particularly, we support this statement by defining and calculating the inflated transfer regions between modalities. Ultimately, this is a major benefit for compliance in robot manipulation.

Although we showcase novel capabilities, there are some limitations. First, hand design plays a crucial role in our manipulation capabilities. We conceptualize an altered hand that would enable rotation about the  $y$ -axis, thus shortening trajectories. While our method may have relied less on data and/or advanced sensing, we note that our manipulation actions were significantly slower than some other related works [11, 75]. Moreover, we discuss the “inflation” parameter  $\rho$  quite extensively, which begs for further theoretical investigation. To this end, the development of geometry-focused modal actions and transitions is an interesting avenue.

Lastly, the current tracking schema requires an object CAD model for training. We are interested in combining object-agnostic tracking methods [93] together with the current object-agnostic controller for instant application to novel objects in the future.



## Chapter 5

# Robot Assembly with Compliant Robots

### 5.1 Introduction

The three previous chapters focused on methods of how we can observe, control, and plan compliant in-hand manipulation online. While these studies were thorough, they typically only discuss a subset of the robot manipulation tasks theorized, i.e., in-hand manipulation. Thus, we are interested in how we can extend this to a wider array of tasks and determine how our formulations can serve for as a foundation for the study of general, whole-arm assembly tasks for robots.

Particularly, vision-based control for robots is a promising approach for traditionally difficult problems. Similar to the motivation for in-hand manipulation, it simplifies the complexity of the system in many capacities by disregarding the need for complex tactile sensors. If compliance can continue to “take up the slack” in control, planning, and modeling uncertainty, it could similarly help aid in achieving task success for general assembly.

In this chapter, we study this concept through two different lenses. First, we attempt to complete difficult tight tolerance insertion and open-world assembly tasks solely with vision. Thereafter, we investigate the benefits of adding minimal force feedback into the control loop as to potentially enable more advanced capabilities.

## 5.2 Vision-based Tight Tolerance Insertion

### 5.2.1 Introduction

Developing robotic systems capable of operating in contact-rich environments with tight tolerances has remained an open research challenge. Despite progress, this is especially true for precision manipulation, where an object must be perceived, grasped, manipulated, and then appropriately placed given task requirements [94–96]. Such functionality is common for everyday insertion tasks e.g., stacking cups into one another, placing a key into a lock, or packing boxes, which are skills particularly advantageous to investigate for developing more capable robots in various application domains [4, 97].

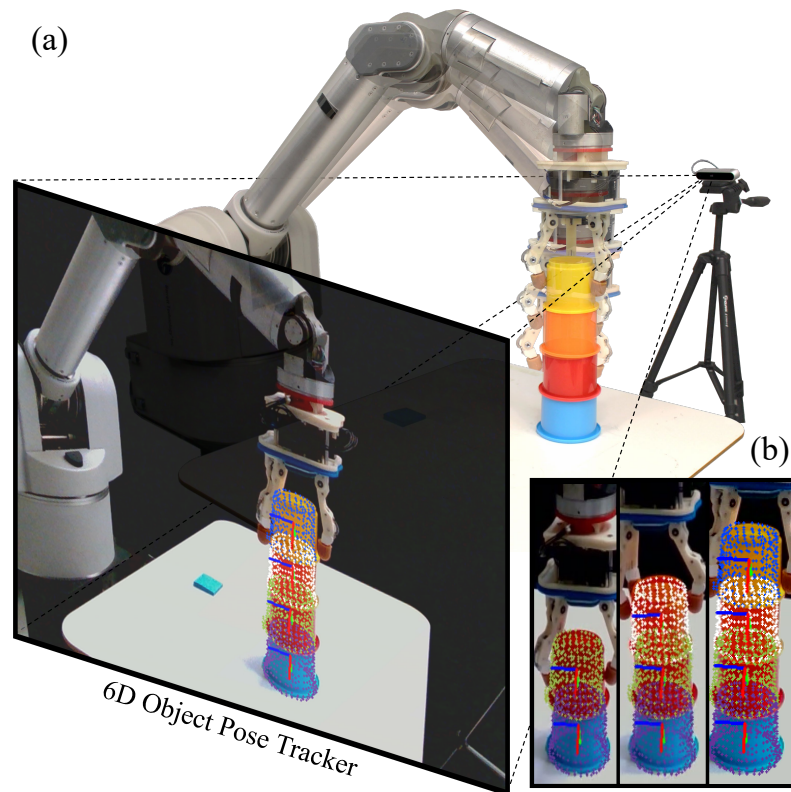


Figure 5.1: (a) An RGBD-based 6D object pose tracker monitors the task state, serving as the primary sensing modality for the robot performing a variety of insertion tasks with tight tolerances, (b) the sequence of cup stacking.

A robotic system’s compliance, i.e., the adaptability of its kinematic configuration, has been key in dealing with uncertainty. Failures during manipulation are often attributed to uncertainty introduced in internal modeling, sensor readings, or robot state, which makes

the system act undesirably in contact-rich scenarios, such as peg-in-hole or assembly tasks. Mechanical compliance, unlike its algorithmic counterpart [28,98], is inherent to the robot’s design and enables the system to passively reconfigure when external contact is enacted [19,99]. Although beneficial, the robot typically lacks otherwise required sensing modalities for precision control, i.e., joint encoders and force sensors can be absent. This poses a limitation, as there exists little knowledge about the robot’s true state [100–102], which complicates the achievement of sub-*mm* accuracy.

Estimating the task’s state dynamically, such as the object’s 6D pose [77,103,104], when onboard sensing is unavailable can be achieved through feedback from inexpensive, external alternatives, such as RGBD cameras [34,105]. Although requiring additional computation, this sensing modality is advantageous as it does not require invasive, bulky sensor suites on the robot, while also providing a wider “field of sensing” for perceiving an extended workspace.

This work investigates whether a framework for compliant systems using exclusively visual feedback can perform precision manipulation tasks. It showcases a complete system and conducts numerous peg-in-hole insertions of varied geometries, in addition to completing several open world constrained placement tasks. In this way, it tests the boundaries of what is possible for vision-driven, compliance-enabled robot assembly, focusing on three main goals: 1) Perform tight tolerance insertion tasks without force sensing, a precise manipulator, or task-specific, online learning; 2) Leverage the extended workspace afforded by compliant, within-hand manipulation to increase the reliability of insertion; and, 3) Demonstrate the role of vision-based feedback for tight tolerance tasks with system compliance. To accomplish these goals, this effort brings together the following components:

- **Vision-based Object Pose Tracker:** Utilizes state-of-the-art low-latency RGBD-based 6D object pose tracking [77], which is shown to be accurate enough when combined with mechanical compliance to solve the target tasks. The visual tracking monitors the task’s state in the form of the object’s 6D pose in a way that is robust to occlusions and varied lighting conditions, while trained solely offline on simulated data (Sec. 5.2.3).

- **Object-Agnostic Within-Hand Manipulation:** Uses a learned model of the inverse system dynamics of an open source and underactuated dexterous hand that is object agnostic [106]. The model is used to perform within-hand manipulation for object orientation alignment that facilitates insertion (Sec. 5.2.3).
- **Visual Feedback Controller:** Develops an insertion control strategy that relies exclusively on the task’s state, which closes the loop through the object tracker’s 6D pose estimate and generalizes to objects of differing geometries so that it is applicable to different scenarios (Sec. 5.2.4). The controller intentionally leverages contacts with the environment and compliance in order to increase reliability, similar to the notion of extrinsic dexterity [107].

This chapter section evaluates the efficacy of the complete system by performing a variety of high-precision manipulation tasks. The results in Sec. 5.2.5 showcase that the proposed general object insertion algorithm, which relaxes rigid insertion constraints due to compliance, exhibits a high rate of success for tight tolerance applications. There are demonstrations indicating that the system further generalizes to a variety of everyday precision placement tasks – stacking cups, plugging a cord into an outlet, inserting a marker into a holder, and packing boxes – underscoring the system’s practical use for complex manipulation scenarios.

## 5.2.2 Related Work

Strategies for peg-in-hole insertion, or more generally robot assembly tasks, have been studied from numerous viewpoints for several decades raising many challenges [94]. Insertion strategies for pegs of various geometries have been previously studied and attempted via a variety of possible techniques and system models: standard cylinders [108–110], multiple-peg objects [111,112], soft pegs [113], industrial inserts [114], and open world objects [115–119]. This work seeks to generalize insertion for various geometries.

### **Model-Based Insertion**

Approaches using contact models reason about state conditions and optimal insertion trajectories while controlling the manipulator [112,113]. These techniques typically require expensive force-torque sensing to detect peg-hole interactions [109,120]. Visual-based methods have been less successful, as estimation uncertainty often causes the manipulator to apply unwanted forces either damaging the robot or its environment [121].

Compliance-enabled architectures, both hardware-based [115, 117] and software-based [108], are widely used to overcome uncertainties in modeling, sensing, and control of a purely rigid system. Such solutions can be applied to the manipulator, the end effector, or both. The majority of previous peg-in-hole insertion works investigate compliance in the manipulator’s control – fixing the object directly to the manipulator and evaluating trajectory search spaces or force signatures [108, 120]. Principally, these works do not address robot assembly tasks, as an end effector is vital for the completion of a pick-and-place style objective. Few works have previously investigated the role of within-hand manipulation for peg-in-hole [122], finding that compliance in a dexterous hand extends the workspace for task completion success. This prior work utilizes a rigid hand with tactile sensors and impedance control, which is computationally inefficient compared to the proposed framework.

### **Learning-Based Insertion**

Regardless of whether a system is rigid or compliant, developing the control policies associated with tight tolerance tasks remain difficult [121]. To address this, learning has been widely performed on such systems – either collecting data through robot interactions in its environment or from human-in-the-loop demonstrations [110]. Reinforcement learning and self-supervised learning [123, 124] have been popular, as they enable optimal policy acquisition without manual data intervention. Conceptually, this feature can be favorable for manipulation learning, as it allows the robot to sufficiently explore its environment and collect representative data of its interactions [114, 116, 125, 126]. But it is also time consuming, computationally expensive, and the extensive number of interactions needed to learn

in a physical environment increases the likelihood of robot damage. It is thus advantageous to develop planning and control techniques that do not require task-specific learning or numerous physical interactions, allowing the system to more easily generalize to novel tasks and environments, as the method proposed here.

### **Visual Feedback Closed-loop Manipulation**

Tremendous progress has been made to adapt control policies reactively with sensing feedback. With recent advances in deep learning, a number of prior works learn a visuomotor controller either by directly mapping raw image observations to control policies [11,127–129] or by using reinforcement learning based on learned visual dynamics models [130–133]. However, this method usually requires a nontrivial amount of training data which is costly to collect in the real world, struggles to transfer to new scenarios, and suffers from the curse of under-modeling or modeling uncertainties [134]. Our work shares the spirit of another line of research that decouples the system into individual sensing and planning components. In particular, [135] developed a compliant manipulation system by integrating a 6D object pose tracker and a reactive motion planner. To generalize to objects of unknown shapes, [136] presents a system that maintains a dual representation of the unseen object’s shape through visual tracking, achieving constrained placement. Although promising results of manipulating one or a few objects have been shown, their scalability to other objects or precision placement tasks remains unclear. In [137], both vision and torque feedback were used for tight insertion. Our work aims instead to robustly tackle a wide range of high precision placement tasks by leveraging the synergy between mechanical compliance and visual feedback, without force or torque feedback.

### **5.2.3 Proposed Framework**

The proposed framework integrates the following components to complete tight tolerance and open world insertion tasks: a 6D object pose tracker given RGBD data, and a compliance-enabled insertion algorithm for a passively compliant arm and dexterous hand. As depicted in the system pipeline (Fig. 5.2), during manipulation, the object tracker (red) asynchronously estimates object poses in the task space, i.e., both the peg and the hole, based

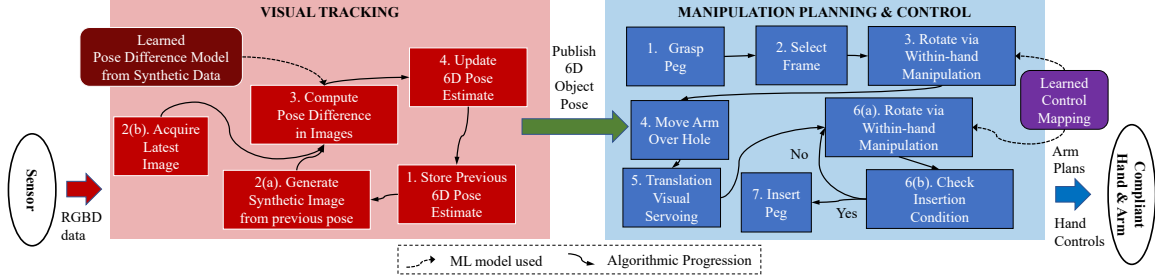


Figure 5.2: System pipeline: (red) a visual tracking framework trained solely with synthetic data to estimate 6D pose differences, provides feedback for (blue) manipulation planning and control of a low-impedance manipulator and a compliant end-effector performing within-hand manipulation for insertion tasks.

on a learned pose difference model, and provides feedback for the insertion planning and control algorithm (blue) to compute action decisions based on an arm motion planner and a learned control mapping of the hand. The following subsections describe these components in more detail.

### Visual 6D Object Pose Tracking based on Synthetic Training

Uncertainty about the task’s state can arise from multiple sources in the target application: (a) the compliant robot arm, (b) the adaptive hand, (c) the potentially occluded object, and (d) the location of the hole. Therefore, dynamic reasoning about the spatial relationship between the peg and the hole is required to achieve reliable tight insertion. This work leverages recent advances in visual tracking that employ temporal cues to dynamically update the 6D pose of tracked objects. In particular, recent work in visual tracking [77] achieves robust and accurate enough estimates at a low latency to work with a wide range of objects. This allows easy integration of visual tracking with planning and control for closing the feedback loop. Additionally, it is also possible to disambiguate the 6D pose of geometrically symmetric objects from semantic textures, thanks to jointly reasoning over RGB and depth data, enabling the overall system to perform a wider range of tasks. Alternative 6D pose tracking methods, which are based solely on depth data [138, 139], often struggle with this aspect. For instance, the green charger utilized later in experimentation exhibits a  $180^\circ$  shape symmetry from the top-down view, whereas only one of the orientations can result in successfully plugging it into a power strip.

The tracker requires access to a CAD model of the manipulated object for training purposes but does not use any manually annotated data. It generalizes from synthetic data to real-world manipulation scenarios. CAD models for this purpose can be imperfect and obtained through inexpensive depth scanning processes [140] or from online CAD libraries. The dimensional error of the CAD model can be larger than that of the tight insertion task considered – given the adaptability afforded to the system via compliance (see plug insertion in Sec. 5.2.5). At time  $t$ , the visual tracker operates over a pair of RGBD images  $I_t$  and  $I_{t-1}$  and predicts the relative pose of the tracked objects parametrized by a Lie Algebra representation  $\Delta\xi \in se(3)$ . The 6D object pose in the camera’s frame is then recovered by  $T_t = exp(\Delta\xi)T_{t-1}$ . During both training and testing,  $I_{t-1}$  is a synthetically rendered image given the pose  $T_{t-1}$  and the CAD model. During training,  $I_t$  is also synthetic but it is the real image during online operation of the system. Thus, a sim-to-real domain shift exists only for the current  $I_t$  image.

The synthetic data generation process is physics-aware and aims for generalization and high-fidelity by leveraging domain randomization techniques [141], as shown in Figs. 5.3 and 5.4. To do so, external lighting positions, intensity, and color are randomized. The training process includes distractor objects from the YCB dataset [142] beyond the targeted objects for manipulation, which introduce occlusions and a noisy background. Object poses are randomly initialized and perturbed by physics simulation with random gravity directions until no collision or penetrations occurs. Background wall textures are randomly selected [143] for each rendering. This rendered image serves as the frame  $I_t$ . In order to generate the paired input image  $I_{t-1}$  for the network, the process randomly samples a Gaussian relative motion transformation  $T_{t-1}^t \in SE(3)$  centered on the identity relative transform to render the prior frame  $I_{t-1}$ . During training, data augmentations involving random HSV shift, Gaussian noise, Gaussian blur, and depth-sensing corruption are applied to the RGB and depth data in frame  $I_t$ , following bi-directional domain alignment techniques [77]. Training on synthetic data takes 250 epochs and is readily applicable to real world scenarios without fine-tuning.

During the task execution, the proposed process tracks all manipulated objects at  $30Hz$ , and provides 6D poses for the planning and control module (Fig. 5.2). Before grasping, the





Figure 5.3: Physics-aware, high-fidelity synthetic training data are augmented via domain randomization.

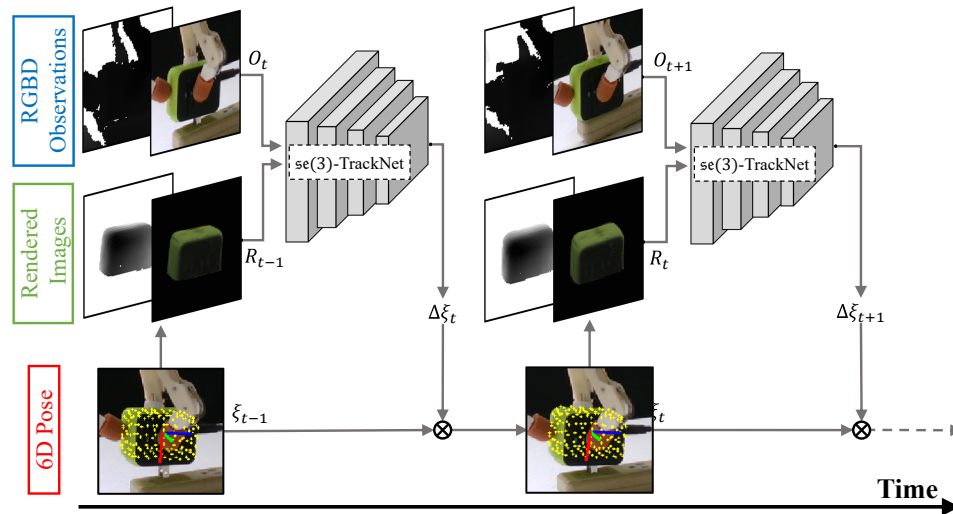


Figure 5.4: 6D pose tracking on RGBD image observations streamed from the camera.

initial 6D pose of each object on the support surface is estimated once via *RANSAC*-based plane fitting and removal [144], followed by a single-shot pose estimation approach [78] to initialize tracking. The tracking process is robust to occlusions and does not require pose re-initialization during manipulation, which is continuously executed until task completion.

### Learning Object-Agnostic Within-hand Manipulation

The employed within-hand manipulation model of a 3-fingered, tendon-driven underactuated hand described is object-agnostic and follows a similar representation to Chapter 3. It relies solely on the reference velocities of the object to suggest manipulation actions [106]. This learned model is realized by first generalizing grasp geometry – the object is represented according to the relative pose of the fingertip contacts after a grasp is acquired. From these contacts, it is possible to strictly define an object frame,  $\mathcal{X}$ , according to Gram-Schmidt orthogonalization. More formally, given  $k$  points of contact,  $\mathcal{P} = p_1, \dots, p_k$  where  $p_i \in \mathbb{R}^3, \forall i \in \{1, \dots, k\}$  between the fingertips and the object with respect to the hand frame, we use  $\mathcal{P}$  to calculate  $\mathcal{X}$  in a closed form [70]. Notably, the relative position of contacts in  $\mathcal{P}$  can sufficiently represent the local geometry of the object in its manipulation plane. From this observation, it is possible to calculate the contact triangle relationship, or distance between the fingertips,

$$\mathcal{T} = (\|p_1 - p_2\|_2, \|p_2 - p_3\|_2, \|p_3 - p_1\|_2) \in \mathbb{R}^3 \quad (5.1)$$

where  $\mathcal{T} = (\mathcal{T}_1, \mathcal{T}_2, \mathcal{T}_3)$  (Fig. 5.5). Note that this representation generalizes object geometry, but not necessarily object dynamics, as the global geometry,  $\Gamma$ , and associated inertia terms of the object are disregarded for simplicity.

Given the object frame  $\mathcal{X}_t \in SE(3)$  at time  $t$ , it is possible to model the configuration, and thus the next-state object frame  $\mathcal{X}_{t+1}$  of an underactuated system given an actuation velocity  $\dot{a}$  based on the system’s energy. With the hand’s joint configuration,  $q \in \mathbb{R}^{\sum_{i=1}^k j_i}$ , which has  $j_i$  joints per finger, the system’s equilibrated joint configuration  $q^*$  can be calculated such that the sum of potential energies between the fingers is minimized. As in any mechanically compliant mechanism, an underactuated system’s degrees of actuation

are fewer than that of its configuration, i.e.,  $\dim(a) < \dim(q)$ . For the adaptive hand, the fingers are actuated via a tendon transmission routed through the joints, providing a constraint:

$$r_{ai}\dot{a}_i = r_{pi}\dot{q}_{pi} + r_{di}\dot{q}_{di}, \quad (5.2)$$

where  $r_{ai}, r_{pi}, r_{di}$  represent pulley radii of the actuator and joints in finger  $i$ , respectively, and  $\dot{a}_i, \dot{q}_{pi}, \dot{q}_{di}$  are the velocities of the actuator and joints, respectively (Fig. 5.5). Given this tendon constraint, and while ensuring  $\mathcal{T}_t = \mathcal{T}_{t+1}$ , i.e., by maintaining integrity on the contact triangle relationship, it is possible to solve for the equilibrated joint configuration:

$$q^* = \arg \min_q \sum_i E^i(q^i) \text{ s.t. } (5.1), (5.2), \quad (5.3)$$

where  $E^i$  represents the potential energy in the  $i^{\text{th}}$  finger:

$$E^i(q^i) = \frac{1}{2}(k_p q_{pi}^2 + k_d q_{di}^2). \quad (5.4)$$

Through Eq. (5.3), which has been shown to successfully transfer physically to an underactuated hand [106], this work efficiently generates system dynamics data by varying relationships in  $\mathcal{T}$  and providing random actuation velocities  $\dot{a}$  to the hand. In doing so, we fill a buffer of 200k object transitions,  $(\mathcal{X}_t, \dot{a}) \rightarrow \mathcal{X}_{t+1}$ , from 50 contact triangle relationships. By taking the element-wise difference of  $\mathcal{X}_t$  and  $\mathcal{X}_{t+1}$ , we calculate  $\dot{\mathcal{X}} \in se(3)$ . Given these action-reaction pairs, we train a fully-connected network to compute the model, or partially constrained Jacobian:

$$g : (\dot{\mathcal{X}}_x, \dot{\mathcal{X}}_y) \rightarrow \dot{a} \quad (5.5)$$

that maps the desired rotational velocity of the object about the  $x$ - and  $y$ -axes to an actuation velocity of the hand,  $\dot{a} \in \mathbb{R}^3$ .

#### 5.2.4 Insertion Strategy

Assume the geometry of the peg,  $\Gamma$ , consists of two opposing, parallel faces with a continuous or discrete set of sidewalls connecting them, where, for the set of all antipodal point contacts

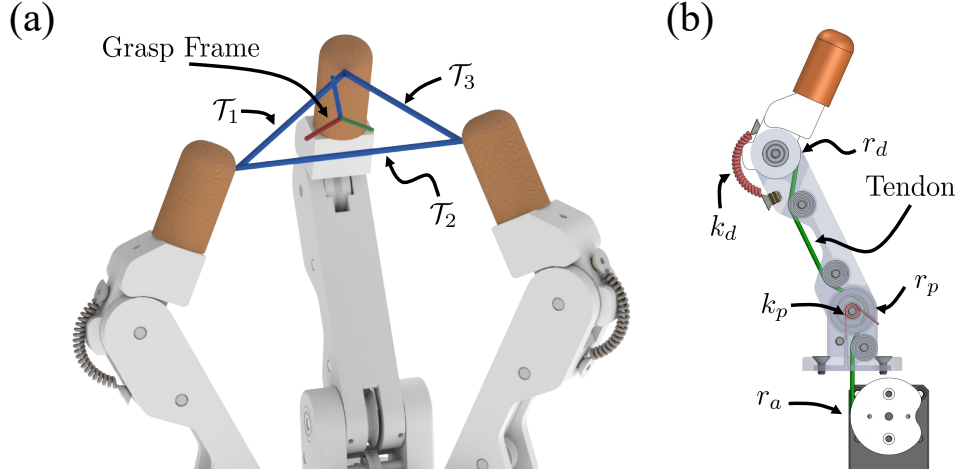


Figure 5.5: (a) Object geometry can be generalized by its resultant contact triangle relationship,  $\mathcal{T}$ . (b) The response of a tendon-driven underactuated finger given actuation is dependent on spring constants and pulley radii.

on the object’s sidewalls, each pair in the set has parallel contact normal force vectors. Conceptual examples of  $\Gamma$  could include standard cylinders or triangular prisms. This section analyzes spatial peg insertion as a planar problem as depicted in Fig. 5.6, i.e., along cross sections of the object. This approach leverages the notion that in practice we can rely on compliance and closed loop control to account for any out of plane misalignment as described in Sec. 5.2.4.

### The Condition for Object Insertion

Consider that a rigid peg is grasped such that its antipodal, i.e., set of opposing, contacts are distanced  $d_o$  from one another across the width of the object. Moreover, the location of the contacts defines the height,  $h$ , of the grasp. Given  $d_o$  and  $h$ , it is possible to define the object’s grasp frame,  $\mathcal{X} \in SE(2)$ , with a pose determined by the grasp-contact vector. The task goal is to insert a peg into its corresponding rigid hole,  $\mathcal{H} \in SE(2)$ , which is parameterized by a width of  $d_h$ . Given  $\Gamma$ , it is possible to define a manipulation frame,  $\mathcal{M} \in SE(2)$ , defined by a transformation,  $T$ , from  $\mathcal{X}$ , that acts as the initial controlled frame for object insertion. Determining  $\mathcal{M}$  for generalized geometries is achieved through PCA, as discussed in the accompanying Appendix A.

In the context of insertion tasks, this work investigates the role of object rotation via

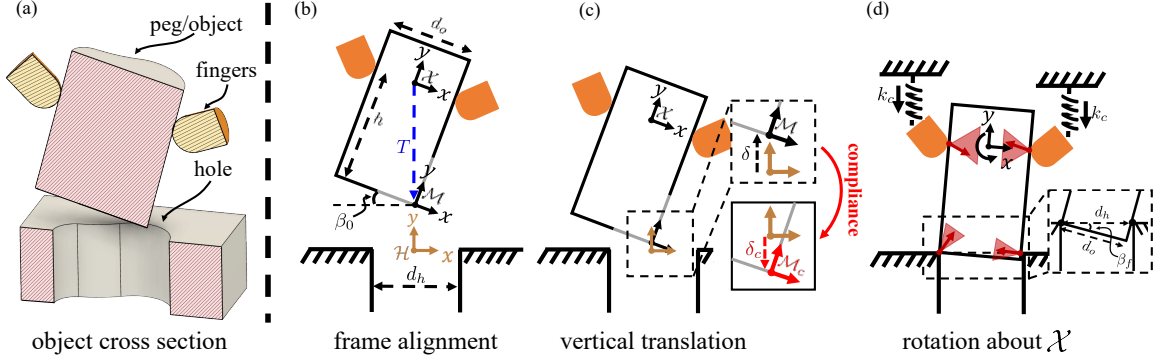


Figure 5.6: (a) Peg insertion is viewed as a planar challenge. (b) For insertion via purely rotational motion about  $\mathcal{X}$ , the peg’s edge is aligned directly above the hole with a starting angle,  $\beta_0$ . (c) Translating  $\mathcal{M}$  downward to  $\delta$  guarantees that a pure rotation will align the peg with the hole, where  $\delta_c$  encourages premature contact and leverages compliance to aid in alignment. (d) Upon rotation about  $\mathcal{X}$ , the peg must overcome contact constraints (red friction cones) of the hole contacts to align for insertion, while aided by virtual spring forces  $k_c$  supplied by compliance.

within-hand manipulation while keeping the relative translational pose between the object frame and the hole fixed. To do so, the process vertically aligns  $\mathcal{H}$ ,  $\mathcal{M}$ , and  $\mathcal{X}$  such that the horizontal component of  $T$  is equal to 0. This therefore sufficiently defines an initial object rotation and sets up spatial peg insertion as a planar problem with a starting angle:

$$\beta_0 = \tan^{-1}\left(\frac{d_o}{2h}\right) \quad (5.6)$$

to satisfy the alignment constraint. It is possible to start in a different angle than  $\beta_0$ , but the position of  $\mathcal{M}$  will need to change accordingly. Once  $\beta_0$  is achieved, the system controls a vertical displacement,  $\delta$ , between  $\mathcal{M}$  and  $\mathcal{H}$ , which can be calculated in closed form such that a pure rotation about  $\mathcal{X}$  places the object in a state aligned with the hole. This final angle of alignment,  $\beta_f$ , can be calculated as:

$$|\beta_f| \leq \cos^{-1}\left(\frac{d_o}{d_h}\right), \quad (5.7)$$

which is dependent on the peg’s two-contact case (Fig. 5.6.d). Thus, the initial insertion

height  $\delta$  before rotation is given as

$$\delta = h (\cos(\beta_0) + \cos(\beta_f)) + \frac{d_o}{2} (\sin(\beta_0) + \sin(\beta_f)). \quad (5.8)$$

The sequential progression of steps for rotation-based object insertion are outlined in Fig. 5.6. In summary: 1) Rotate  $\mathcal{M}$  such that it aligns with  $\mathcal{H}$ ; 2) Translate  $\mathcal{M}$  downward to  $\delta$ ; and, 3) Rotate about  $\mathcal{X}$  until the two-contact case is overcome, i.e., rotation of  $\mathcal{X} < |\beta_f|$ , and the peg is aligned with the hole. Once satisfied, the remaining actions for insertion seek object translation downward while avoiding jamming along the sidewalls. To account for this, this work leverages compliance and controlled object spiral motions [120] provided by within-hand manipulation.

### On the Role of Compliance

While the above formulation holds for rigid insertion in the relaxed planar representation, the bounds of  $\delta$  are extremely small for tight tolerance cases ( $< 1.5mm$  if perfectly aligned). Moreover, when considering any out-of-plane misalignment in the true additional dimension that is not modeled, the peg can easily become off-centered with respect to the hole. The features inherent to compliance benefit this task, as it not only enables adaptability to the out-of-plane misalignment, but also allows to relax these strict bounds of  $\delta$  by developing a compliance-enabled insertion distance,  $\delta_c$  (Fig. 5.6.c). This deviation in depth is nontrivial to determine in its closed form – as it depends on models of the manipulator-hand system’s compliance, the forces that can be applied to the object via the hand, and an approximation of the nonlinear contact dynamics associated with peg-hole interactions (Fig. 5.6.d). Although difficult to model, a properly tuned translational deviation in  $\delta_c$  towards the hole from  $\delta$  is advantageous, as it encourages contact between the hole and the object prematurely so that the contact constraints encourage and assist in alignment before insertion.

## Insertion Algorithm

The proposed system is controlled by closing the loop through visual feedback. In real time, the visual tracker monitors the state of the task space; specifically, the relative pose between the peg and the hole. This relationship is used to adjust the control reference setpoints of both the manipulator’s joints and the tendons in the hand, regardless of their believed configuration states. By continually servoing the object’s  $SE(3)$  pose relative to the hole, it is possible to complete tight insertion tasks while adapting to external disturbances and noisy pose estimates (Sec. 5.2.5).

**Vision-Driven Object Insertion:** The insertion sequence shown in Alg. 7 begins with the object tracker asynchronously monitoring the pose of both, the object and the hole. Once the object is grasped off of the support surface, the precomputed transformation,  $T$ , via PCA (A.1), from the manipulation frame,  $\mathcal{M}$ , can be rigidly attached to  $\mathcal{X}$ .

---

### Algorithm 7 Vision-Driven Object Insertion

---

**Input:**  $\Gamma, (\beta_0, \beta_f, \delta_c, \gamma, \sigma)$   $\triangleright$  object geometry, *hyparms*(initial object angle, final object angle, insertion depth, alignment tolerance, step length)

- 1:  $T, \pi_1 \leftarrow PCA(\Gamma)$   $\triangleright$  POM transform, principal axis (A.1)
- 2: **tracker.start\_async\_perceive**( $\Gamma$ )  $\triangleright$  start tracking thread
- 3: **tracker.attach\_transform**( $T$ )  $\triangleright$  attach  $T$  to  $\mathcal{X}$  for  $\mathcal{M}$
- 4:  $\mathcal{X}, \mathcal{H} \leftarrow \mathbf{tracker.get_poses}()$   $\triangleright$  object & hole 6D poses
- 5: **system.grasp**( $\mathcal{X}, \pi_1$ )  $\triangleright$  grasp and lift object
- 6: **hand.rotation\_servo**(**tracker**,  $\pi_1, \beta_0$ )  $\triangleright$  Alg. 8
- 7:  $\mathcal{M}, \mathcal{H} \leftarrow \mathbf{tracker.get_poses}()$
- 8: **arm.move\_above**( $\mathcal{M}, \mathcal{H}$ )  $\triangleright$  place edge above hole
- 9: **arm.translation\_servo**(**tracker**,  $\delta_c, \gamma, \sigma$ )  $\triangleright$  Alg. 9
- 10: **hand.rotation\_servo**(**tracker**,  $\pi_1, \beta_f$ )  $\triangleright$  Alg. 8
- 11: **system.spiral\_insertion**()  $\triangleright$  coordinated spiral insertion

---

Upon doing so, within-hand manipulation is performed such that the initial insertion angle,  $\beta_0$  from Sec. 5.2.4 is achieved along the principal axis,  $\pi_1$  (Alg. 8). The velocity references,  $\dot{\mathcal{X}}_x$  and  $\dot{\mathcal{X}}_y$ , during this within-hand manipulation process are chosen so as to minimize any rotation not along this principal axis. Upon reaching  $\beta_0$ , the robot arm plans a trajectory such that the resultant position of the manipulation frame is directly above the hole. Due to the robot’s imprecision, however, the desired pose of  $\mathcal{M}$  above  $\mathcal{H}$  is often not accurately achieved.

At this point, the *translation\_servo* method begins by sequentially adjusting the control reference of the manipulator based on feedback from the object tracker (Alg. 9). Specifically, given the hyper-parameter  $\gamma$ , if the hole  $xy$ -plane translational difference between  $\mathcal{M}$  and  $\mathcal{H}$  is within  $\gamma$ , the manipulator will move a step,  $\sigma$ , which can be fixed or adaptive, downward towards the hole by adjusting the joint target values of the robot. Otherwise, the robot will move in the Cartesian step,  $\sigma$ , towards aligning  $\mathcal{M}$  with  $\mathcal{H}$ . This process is continually repeated until the vertical threshold,  $\delta_c$ , is reached.

---

**Algorithm 8** Within-Hand Rotation Visual Feedback Control

---

**Input:** `tracker`,  $\pi_1$ ,  $\theta \in \{\beta_0, \beta_f\}$

		▷ principal axis of rotation, object target angle
	State $\mathcal{X}, \mathcal{H} \leftarrow \text{tracker.get\_poses}()$	▷ object and hole 6D pose
1:	$R_\Delta \leftarrow \mathcal{X}.R - \mathcal{H}.R$	▷ relative rotation $\in SO(3)$
2:	<b>if</b> $\theta$ <b>is</b> $\beta_0$ <b>then</b>	▷ increase angle for edge insertion
3:	<b>while</b> $\mathcal{H}.R(\pi_1) - \mathcal{X}.R(\pi_1) \leq \beta_0$ <b>do</b>	
4:	$\dot{\mathcal{X}} \leftarrow [-R_\Delta(\pi_{1x}), -R_\Delta(\pi_{1y})]$	▷ rotate along $\pi_1$
5:	$\dot{a} \leftarrow \text{hand.model.predict}(\dot{\mathcal{X}})$	▷ Eq. (5.5)
6:	<b>hand.actuate</b> ( $\dot{a}$ )	▷ send action to motors
7:	$\mathcal{X}, \mathcal{H} \leftarrow \text{tracker.get\_poses}()$	
8:	$R_\Delta \leftarrow \mathcal{X}.R - \mathcal{H}.R$	
9:	<b>if</b> $\theta$ <b>is</b> $\beta_f$ <b>then</b>	▷ decrease angle, align with hole
10:	<b>while</b> $R_{\Delta x} \geq \beta_f$ <b>or</b> $R_{\Delta y} \geq \beta_f$ <b>do</b>	
11:	$\dot{\mathcal{X}} \leftarrow [R_{\Delta x}, R_{\Delta y}]$	
12:	$\dot{a} \leftarrow \text{hand.model.predict}(\dot{\mathcal{X}})$	
13:	<b>hand.actuate</b> ( $\dot{a}$ )	
14:	$\mathcal{X}, \mathcal{H} \leftarrow \text{tracker.get\_poses}()$	
15:	$R_\Delta \leftarrow \mathcal{H}.R - \mathcal{X}.R$	

---

The hand then attempts to reorient  $\mathcal{X}$  with  $\mathcal{H}$  while maintaining a small downward force between the object and the hole’s edges, provided by system compliance and  $\delta_c$ . Orientation alignment of the peg is solely achieved through within-hand manipulation (Alg. 8); where the object no longer follows rotations along  $\pi_1$ , but now by the true rotational difference  $R_\Delta$  between the hole and the peg. This process continues until the rotational angle is less than  $\beta_f$ .

The algorithm concludes by performing a spiral insertion technique along the roll and pitch axes of the object. If the object were unconstrained, the hand’s actuation would provide a spiral pattern of  $\mathcal{M}$ , as in related work [108, 120], while the arm attempts to



slowly translate downward. This coordinated hand/arm motion helps limit jamming and encourages proper insertion until the object is fully seated into the hole.

---

**Algorithm 9** Arm Translation Visual Feedback Control

---

**Input:** `tracker`,  $\delta_c$ ,  $\gamma$ ,  $\sigma$

```

1:  $\mathcal{M}, \mathcal{H} \leftarrow \text{tracker.get\_poses}()$ 
2: while  $\mathcal{M}.t_z - \mathcal{H}.t_z > \delta_c$  do
3:    $\mathcal{E}_{\Delta x}, \mathcal{E}_{\Delta y}, \mathcal{E}_{\Delta z} \leftarrow 0, 0, 0$ 
4:   if  $-\gamma \leq \mathcal{M}.t_x - \mathcal{H}.t_x \leq \gamma$  and
        $-\gamma \leq \mathcal{M}.t_y - \mathcal{H}.t_y \leq \gamma$  then
5:      $\mathcal{E}_{\Delta z} \leftarrow -\sigma$ 
6:   else
7:      $\epsilon_x \leftarrow \mathcal{X}.t_x - \mathcal{H}.t_x$ 
8:      $\epsilon_y \leftarrow \mathcal{X}.t_y - \mathcal{H}.t_y$ 
9:      $\mathcal{E}_{\Delta x}, \mathcal{E}_{\Delta y} \leftarrow -\text{sgn}(\epsilon_x)\sigma, -\text{sgn}(\epsilon_y)\sigma$ 
10:   $\mathcal{A} \leftarrow \text{arm.motion\_planner}(\mathcal{E}_{\Delta})$ 
11:   $\text{arm.execute}(\mathcal{A})$ 
12:   $\mathcal{M}, \mathcal{H} \leftarrow \text{tracker.get\_poses}()$ 

```

▷ insertion depth, alignment tolerance, step length

▷ check translation along z

▷ servo end effector pose downward

▷ servo translation to align with hole

▷ plan motion delta

▷ move end effector

---

### 5.2.5 Experiments

We test the proposed insertion framework using a robotic system comprised of a low-impedance manipulator for object translation, an underactuated hand performing within-hand manipulation for object rotation, and a low latency 6D object pose tracker based on RGBD data (Figs. 5.1, 5.8). The manipulator, a 7-DOF Barrett WAM arm, utilizes the *RRTConnect* algorithm in OMPL [145] and is imprecise due to an inaccurate internal model of its true system dynamics, with translational errors as large as 2.6cm. The end effector, an adapted Yale OpenHand Model O [146], is a mechanically adaptive hand comprised of six joints and three controlled actuators, and is not equipped with joint encoders or tactile sensors. Changes to the open source design include joint bearings to reduce friction, and rounded fingertips to assist in within-hand manipulation. Finally, the 6D object pose tracker, which was calibrated to the robot’s world frame, takes the RGBD images streamed from an external, statically mounted Intel Realsense D415 camera, and in real-time, estimates the 6D pose of the manipulated object to provide feedback for control.

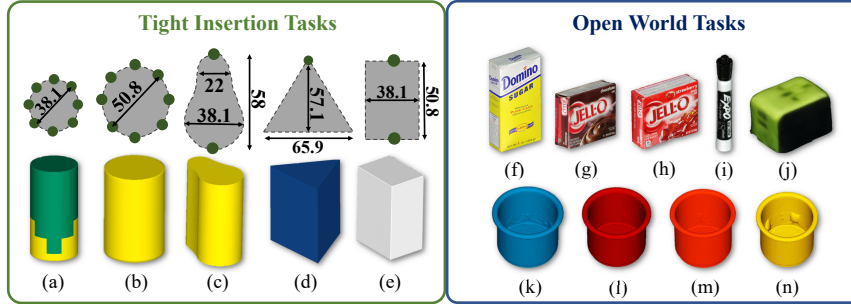


Figure 5.7: Experimental objects considered in the **Tight Tolerance Tasks** and **Open World Tasks**. All lengths are in  $mm$  and points on object faces indicate PCA-determined edge manipulation frames. (a) small circle, (b) large circle, (c) pear, (d) triangle, (e) rectangle, (f) YCB *004\_sugar\_box*, (g) YCB *008\_pudding\_box*, (h) YCB *009\_gelatin\_box*, (i) YCB *040\_large\_marker*, (j) green charger. (k)-(n) YCB *065\_cups*.

The system is tested via numerous insertion tasks. First, five 3D printed peg-in-hole objects were designed with  $< 0.25mm$  hole tolerances and were painted different colors and/or patterns, for evaluation on both textured and textureless objects. These objects are described based on their face geometries: namely, the small circle, large circle, pear, triangle, and rectangle (Fig. 5.7). Insertion was tested for each of these objects individually, where we then sequentially isolate individual system components – compliance, control, and sensing – to evaluate their effects on task success. Finally, we assessed the efficacy of the approach with six open world insertion tasks, involving nine different objects with diverse and challenging properties (textureless, reflective, flat and thin shapes, etc.), to underscore the utility of the framework in complex manipulation scenarios (Fig. 5.8).

### Tight Tolerance Object Insertion

This test involved 12 insertions for each of the five objects using Alg. 7. Upon task reset, objects were placed back onto the support surface in no predefined pose; it was up to the system to initialize and track this pose and reacquire a grasp. The results, presented in Table 5.1, depict the planning time, total execution time, number of hand actions used for within-hand manipulation, and success rate for each object scenario. The large circle had the highest rate of success compared to any of the other objects, while the triangle had the lowest. The two objects that did not have curved edges, i.e., the triangle and the rectangle, were the most difficult out of the five to insert. The interpretation is that the constraints

of the task, i.e., sharp edges of the object’s face, did not enable system compliance to easily align the yaw rotation of the object with the hole. This posed a slight challenge for the relaxed planar insertion strategy from Sec. 5.2.4, and required precise yaw rotation via the manipulator, which was not required for the other objects.

Table 5.1: TIGHT TOLERANCE INSERTION RESULTS

<b>Obj.</b>	<b>Planning (s)</b>	<b>Total (s)</b>	<b>Hand Actions</b>	<b>Success</b>
Small Cir.	$4.1 \pm 0.92$	$94.8 \pm 10.78$	$48.9 \pm 8.76$	<b>10/12</b>
Large Cir.	$3.5 \pm 0.52$	$88.8 \pm 11.54$	$44.9 \pm 9.10$	<b>11/12</b>
Pear	$4.0 \pm 1.05$	$77.9 \pm 10.34$	$35.2 \pm 8.62$	<b>9/12</b>
Triangle	$3.7 \pm 2.08$	$90.9 \pm 14.00$	$27.4 \pm 4.60$	<b>8/12</b>
Rectangle	$6.2 \pm 3.53$	$106.4 \pm 23.43$	$40.9 \pm 5.90$	<b>9/12</b>

### System Analysis

Several ablation studies were performed using three of the five evaluated objects – the large circle, the pear, and the rectangle – in order to better understand how different components of the system contribute to task success. In particular, these tests evaluate the effects of: 1) reducing system compliance via the hand, the arm and the environment; 2) performing insertion with differing levels of feedback, i.e., naive and open loop control; and, 3) deliberately introducing noise into the object tracker’s pose estimation to simulate higher perception uncertainty. (Table 5.2). The Appendix A describes how the system compensated for disturbances.

**Reducing system compliance:** To test the role of compliance for task success, three altered system configurations were developed: 1) A system with a rigid, parallel Yale Open-hand Model GR2 gripper [48] with custom fingertips as to immobilize the object to the end of a low-impedance manipulator; 2) A system with a rigid 3-fingered Robotiq hand affixed to the end of a rigid Kuka IIWA manipulator but allowing compliance in the environment by attaching a standard packing box at the base of the hole; and, 3) The same rigid robot setup with the Robotiq hand and the Kuka manipulator but with a fixed hole, removing any presence of compliance. Fig. A.2 in the Appendix highlights the different variations considered in terms of compliance. Twelve insertions for each of the three test objects were

performed in each case while following Alg. 7. During task execution, all object rotations that were performed by the hand originally, were now controlled solely by the manipulator. Results indicate, as presented in Table 5.2, that compliance at some level in the arm/hand/object system is beneficial for task completion. The worst success rate arises for a fully rigid system, task, and environment. The Kuka setup with a compliant hole performed well, especially for the difficult rectangle insertion task, assisted by the precision of the object tracker and the higher accuracy of the rigid Kuka manipulator. Planning time between systems varied due to differences in computational resources, and thus it is not directly compared.

**Naive and open loop control:** These tests evaluate the effects of relinquishing feedback by limiting the amount of information transferred from the tracker to the insertion algorithm. This is tested both via naive control and an open loop configuration of the system. Naive control attempted to perform the same sequence of actions as in Alg. 7, but did not use any within-hand manipulation of the object after grasping, i.e., visual servoing was purely translational with the manipulator and Alg. 8 was not used. The compliant hand in this case effectively acted as a remote center of compliance [117,147]. In open loop testing, a single plan was computed and executed from the starting grasp configuration up until object insertion without utilizing in-hand manipulation or any visual feedback during the task, i.e., neither Alg. 8 or Alg. 9 were used. Notably, both of these ablations performed very poorly in testing, as the imprecision of the manipulator and lack of a controlled object rotation to aid in insertion, drastically impeded task success (Table 5.2).

**Noisy pose estimate:** The final ablation experiment evaluated how the accuracy of pose estimation played a role in task success. While following the same sequence of actions as in Alg. 7, uniformly sampled noise was introduced into the pose output of the tracker at two different levels. The first test introduced uniform sampled noise within  $5mm$  and  $5^\circ$  to the pose estimate. The second test introduced uniform noise within the range of  $10mm$  and  $10^\circ$ . As indicated by the results in Table 5.2, the system was largely able to compensate for this noise at the  $5mm/5^\circ$  level, successfully completing 7 for the large circle, 4 for the pear, and 4 for the rectangle out of 12 executions. The additional noise from the  $10mm/10^\circ$  test drastically decreased success. The total task time for both noise level increased significantly

Table 5.2: SYSTEM ABLATION ANALYSIS

	Obj.	Planning (s)	Total (s)	Hand Actions	Success
Reducing Compliance	<b>Rigid Hand / Compliant Arm / Rigid Hole</b>				
	Large Cir.	$7.8 \pm 1.65$	$62.3 \pm 8.74$	-	6/12
	Pear	$11.0 \pm 6.76$	$76.4 \pm 28.10$	-	5/12
	Rectangle	$12.6 \pm 4.37$	$92.3 \pm 24.95$	-	3/12
	<b>Rigid Hand / Rigid Arm / Compliant Hole</b>				
	Large Cir.	$1.4 \pm 0.12$	$65.6 \pm 3.53$	-	9/12
	Pear	$1.5 \pm 0.13$	$68.9 \pm 3.92$	-	4/12
	Rectangle	$1.5 \pm 0.12$	$71.3 \pm 4.35$	-	12/12
	<b>Rigid Hand / Rigid Arm / Rigid Hole</b>				
	Large Cir.	$2.2 \pm 0.37$	$88.9 \pm 10.48$	-	6/12
	Pear	$2.2 \pm 0.25$	$91.4 \pm 8.46$	-	0/12
	Rectangle	$2.1 \pm 0.14$	$90.4 \pm 13.09$	-	2/12
Limiting Control	<b>Naive (omit Alg. 8)</b>				
	Large Cir.	$4.8 \pm 1.78$	$78.0 \pm 22.64$	-	2/12
	Pear	$7.7 \pm 1.25$	$81.9 \pm 10.13$	-	0/12
	Rectangle	$8.9 \pm 0.95$	$103.5 \pm 5.78$	-	0/12
	<b>Open Loop (omit Alg. 8 and Alg. 9)</b>				
	Large Cir.	$3.8 \pm 1.09$	$53.47 \pm 9.78$	-	0/12
Pear	$5.1 \pm 1.62$	$61.96 \pm 4.01$	-	0/12	
Rectangle	$5.1 \pm 1.22$	$65.45 \pm 5.18$	-	0/12	
Noisy Sensing	<b>Uniform Noise – 5mm / 5°</b>				
	Large Cir.	$5.9 \pm 0.99$	$127.5 \pm 18.11$	$50.6 \pm 18.23$	7/12
	Pear	$10.7 \pm 2.29$	$138.9 \pm 12.46$	$40.2 \pm 7.99$	4/12
	Rectangle	$9.9 \pm 1.70$	$137.1 \pm 6.28$	$49.7 \pm 4.19$	4/12
	<b>Uniform Noise – 10mm / 10°</b>				
	Large Cir.	$11.7 \pm 2.29$	$148.5 \pm 12.46$	$39.6 \pm 11.19$	1/12
Pear	$12.6 \pm 0.33$	$134.1 \pm 8.82$	$34.9 \pm 7.19$	0/12	
Rectangle	$13.4 \pm 1.97$	$212.6 \pm 13.58$	$51.9 \pm 11.79$	0/12	

as compared to the baseline trials in Sec. 5.2.5. This increase in execution time is attributed to Alg. 9, where noise continually moves  $\mathcal{M}$  outside of the  $\gamma$  threshold insertion region, so the manipulator slowly oscillates back and forth until  $\delta_c$  is achieved. Conclusively, the ability of the algorithm to complete sub- $mm$  accuracy insertion tasks with purposefully added noise indicates the robustness of the overall framework.

### Open World Tasks

Finally, a series of open world tasks were attempted to highlight the utility of the proposed system in complex manipulation scenarios: marker insertion, plug insertion, box packing, and cup stacking (Fig. 5.8(e)). These tests mostly utilize objects contained in the YCB Object and Model Set [142], which provides object CAD models for tracking. The charging plug, however, was 3D scanned using [140] with an inexpensive RGBD sensor, providing a coarse representation of the true object model (Fig. 5.7). The goal for each task was to grasp, manipulate, and insert the object so as to reach its desired hole configuration, which was predefined depending on task requirements. Of the six tasks, three – gelatin box, pudding box, and sugar box – came in the form of packing, where a single box was removed from the case and had to be replaced. Two other tasks – marker insertion and plug insertion – were performed such that the marker was placed into a holder and the plug was inserted into an outlet. The most difficult of the open world tasks corresponded to cup stacking. This task requires sequential tracking of both the cup that is to be stacked on top of and the cup being manipulated. The proposed system is able to complete this task placing four cups successfully on top of one another. These evaluations showcase the tracking and manipulation capabilities of the proposed system, which are also highlighted in the supplementary video.

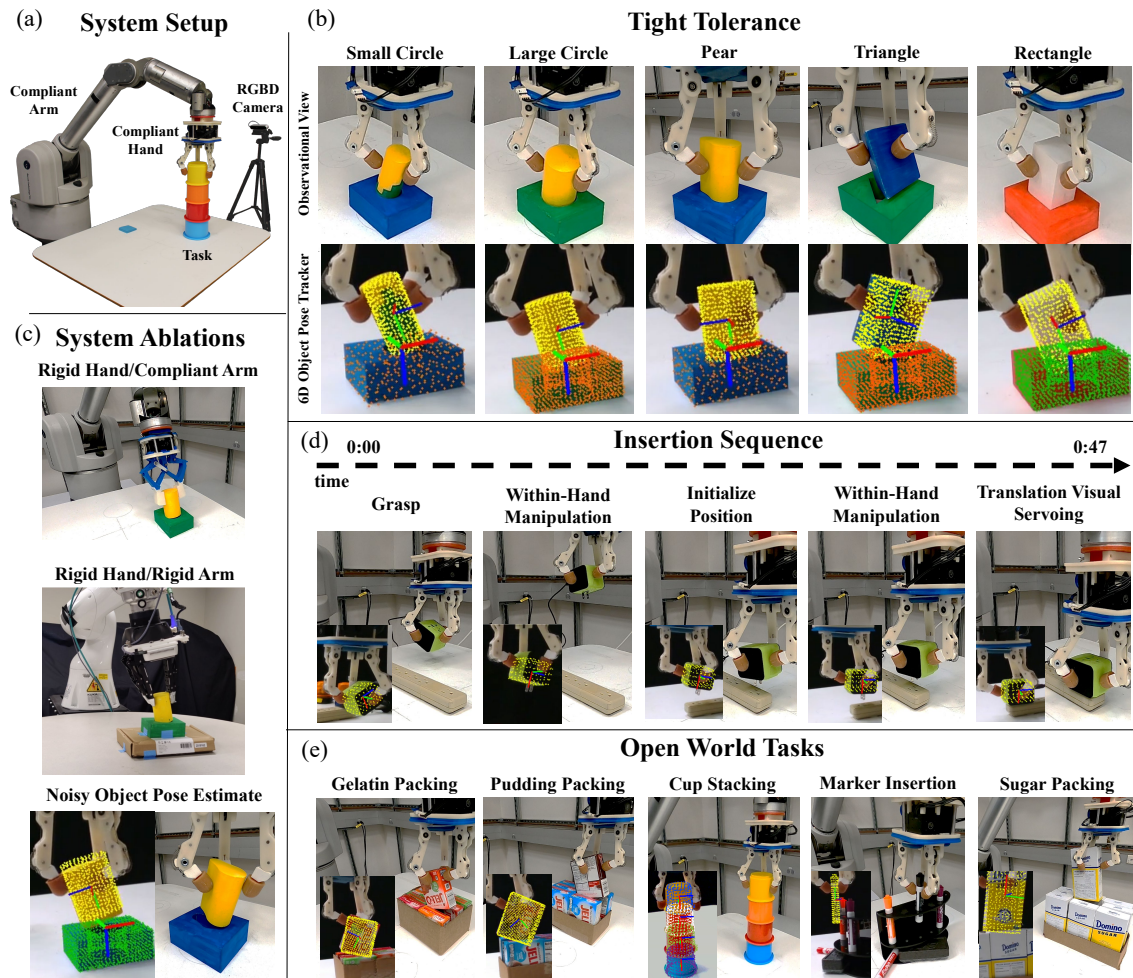


Figure 5.8: (a) System setup overview. (b) Tight tolerance insertion of 5 peg geometries, highlighting the observational/external view and the tracked 6D pose via the RGBD camera. (c) System ablations include reducing compliance of the hand and the arm, in addition to deliberately adding noise to the pose estimate. (d) The open world task of plug insertion is highlighted, showcasing the sequence of actions taken from grasp to insertion. (e) Five other open world tasks were also evaluated – box packing, marker insertion, and cup stacking. Please refer to the supplementary video for a complete overview of evaluated tasks.

### 5.2.6 Discussions

This chapter section presented a vision-driven servoing framework that tackles the problem of controlling compliant, passively adaptive mechanisms for precision manipulation. The framework is able to perform 5 tight tolerance and 6 open world tasks – regardless of whether the object CAD model was imperfect or if pose estimate feedback was noisy (Sec. 5.2.5). In summary, the contributions of this work are fourfold:

- **Precision control with minimal on-board sensing:** The framework can reliably complete an array of insertion tasks – including those with tight tolerances – without force-torque sensors on the manipulator or tactile sensors/joint encoders on the compliant hand.
- **Passively compliant within-hand manipulation for insertion:** The system utilizes controlled hand actions to extend the task’s workspace and provides an added layer of compliance to limit object insertion jamming.
- **Vision-driven feedback controller:** The servoing insertion algorithm is able to generalize to different object geometries, and can be easily utilized for other robot assembly and insertion tasks given compliance.
- **Utilizing the environment to solve the task:** The control strategy intentionally leverages premature contacts by relaxing the rigid constraints of the task, which is possible and effective given compliant systems. In this way, this work applies the principle of ”extrinsic dexterity” [107] in the context of insertion tasks with tight tolerances.

There are several aspects of the proposed system that are worth pursuing in future work: 1) Adapt the insertion algorithm so as to better leverage compliance for complex out-of-plane geometry, such as in cases of sharp edges or non-convex objects; 2) Optimize a passively compliant hand design that is capable of performing both finger gaiting and in-hand manipulation with a large workspace; 3) Develop and integrate a model-free perception tracker into the visual feedback framework to reduce dependence on the object’s CAD model; 4) Incorporate learning into the insertion algorithm so as to first automatically tune any



object-specific hyperparameters and then incrementally develop more effective strategies in a RL fashion; and finally, 5) Increase difficulty of the tasks by performing more advanced and sequential assembly procedures, such as assembling toys or simple pieces of household furniture.

## 5.3 Force-based Tight Tolerance Insertion

### 5.3.1 Introduction

Robots interact with the world through physical contact. Contact has been studied in the literature for decades and can be conceptualized a number of ways, but in its most fundamental sense, contacts are interaction-based constraints imposed on the relative motion between objects. As humans, we often rely on contact to change the state of our environment, whether it be removing keys from a bag or cleaning a dirty dish. While seemingly simple, contact is quite complex in that it requires accurate parameter estimation in order to sufficiently predict its current state, i.e., static, stick-slip, or slip conditions. Previous works have tried to estimate various properties of contact during manipulation, e.g., the location of the contact [148], frictional and curvature properties [47,55], slip conditions [18], etc. but have witnessed many bottlenecks.

Fundamentally, contacts define constraints and constraints define the available degrees-of-freedom (DOF) of an object. In free space, an object is able to move in 6-DOF, but a non-cylindrical peg inserted into a slot is only able to translate in 1-DOF. When there exists some degree of uncertainty in the estimation of contact, the calculated constraints can be non-representative of the system’s true state. For instance, let’s assume there exists little uncertainty in the estimation of point contact locations. Here, we can claim that the restricted motion of an object with two point contacts may be similar or even identical in nature to the constraints imposed on an object with 100 edge point contacts on or near a single axis. In a practical sense, this distinction is insignificant in the free DOFs of the object, and merely adds needless computation. Now, let’s be more practical and assume there does exist some degree of uncertainty in our estimation of contact locations, even in a single point out of the 100. This calculation may estimate additional constraints placed on the object which are not physically valid. Thus, instead of explicitly defining the state of every contact between a robot and its environment, which is quite computationally expensive and is gravely subject to uncertainty, we in this work abstract out individual contact properties and are conversely interested in contact formations, or more generally, the constraints placed on an object’s motion when in contact with its environment.

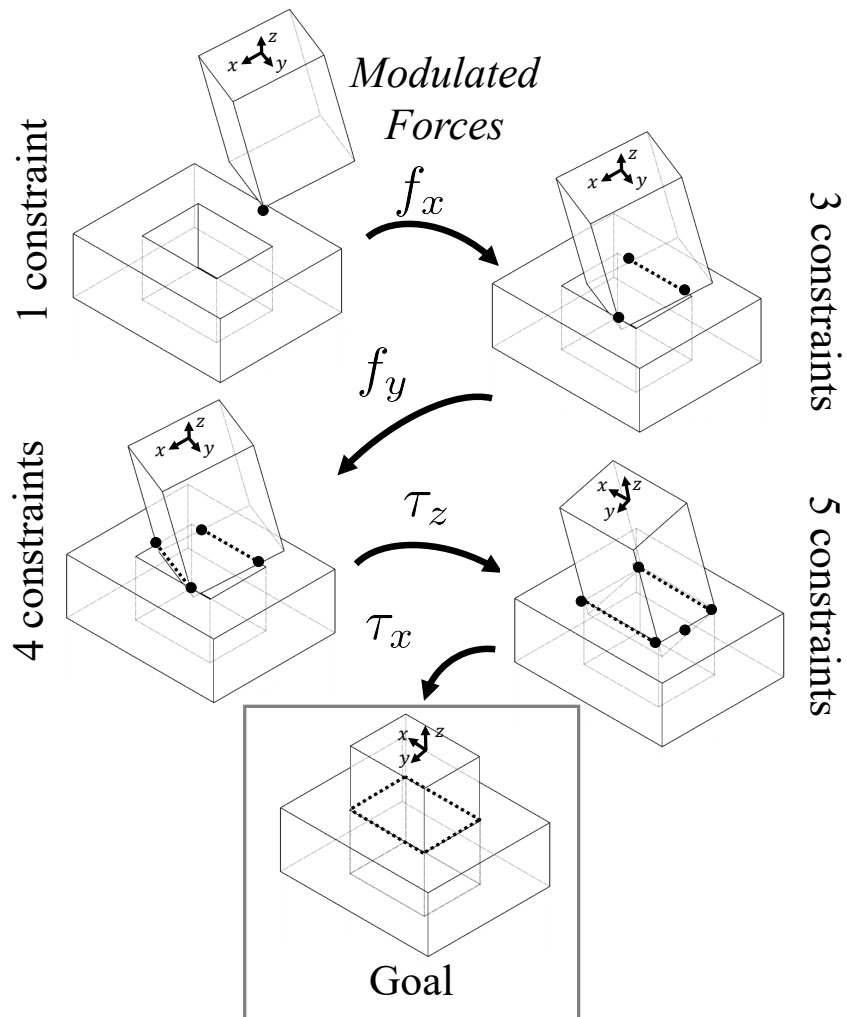


Figure 5.9: Object insertion can be conceptualized as the continual addition and modulation through time of an object's constrained degrees-of-freedom. By continually modulating forces once constraints are detected, tight tolerance insertion can be achieved without *a priori* knowledge of the object geometry or exact hole pose for convex objects.

*Contact formations* can be conceptualized as a grouping of contact sets that constrain an object in similar ways [149]. For instance, various contact scenarios could constrain either some translational DOFs, e.g., sphere in tube, of an object or all translational DOFs, e.g., sphere in case, and the set of all contact groupings that constrain an object in a similar way would be considered in a contact formation. Traditionally, acquiring and maintaining contact formations on a real robot platform was difficult due to an inability to practically modulate forces between two bodies over time. Within the past decade, a great deal of work has addressed the concept of dealing with uncertainty in robot mechanics—specifically, the ability to adapt to external forces as to continually maintain contact with an object. Compliant manipulation frameworks, either in the form of active [108] or passive architectures [115, 117], enable this ability—allowing a system to kinematically adapt to uncertain environments and maintain desired forces. This introduces the idea of *compliance-enabled contact formations*, which are leveraged within this work to maintain a desired type of contact during the task.

Combining these ideas, this work investigates the applicability of controlling tight tolerance insertion tasks via the chaining of contact formations. Specifically, we leverage a control approach that modulates forces in controlled directions, and by continually adding forces along specific axes until contact, i.e., constraints, we transition between different contact formations until the task is successfully completed (Fig. 5.9). This approach utilizes external contact as a means of constraining the object’s potential motion [107], which in turn limits uncertainty of the object’s current configuration. While utilizing a compliant robot, this system aids in both acquiring and maintaining a desired contact configuration. Moreover, we show how in-hole jamming is limited due to this compliance—functioning similarly in concept to a remote center of compliance (RCC) device [150, 151]. Retrofitted with a 6-axis force/torque sensor at the end effector and an in-hand camera, we monitor the state of the object and servo to its desired goal forces and axial alignment. We can thus summarize our contributions:

1. Our algorithm is an object-agnostic procedure that does not require *a priori* knowledge of the object geometry, exact hole position, or exact hole orientation. To our

knowledge, this is one of few works that can insert an object when axial rotation is largely unknown ( $> 40^\circ$ ).

2. We do not require complex analytical or computationally expensive learned models of contact dynamics—our method relies solely on the idea of validating constraints via axial force application through time.
3. This method is conceptually straightforward, as it solely relies on following a desired trajectory in Contact Formation-space, and can thus be run in real-time on a physical robot.
4. We justify the true utility of using compliant systems for tight tolerance, contact-rich tasks by quantifying reconfigurability. Specifically, we quantify slip conditions for our underactuated hand, and show how the hand reorganizes its contacts when excessive forces are applied, which helps eliminate in-hole jamming.

### 5.3.2 Related Work

Generalized robot assembly has been investigated for decades and the development of a practical, holistic solution has faced many challenges [94]. Elucidating this grand challenge can be found in previous works with: standard cylinders [108–110], multiple-peg objects [111, 112], soft or compliant cylinders [113], industrial inserts [114], and standard open world objects [115–119]. In an attempt to extend previous work, we in this work outline a method that underscores the idea of *generalization*, in that our method provides a practical solution to insertion when there exists positional and orientational uncertainty of the hole pose, with minimal knowledge of the object geometry.

#### Peg-in-hole Assembly

Approaches to solving generic assembly tasks can be divided into two different categories: analytical approaches and learning-based approaches. In the former, approaches utilize contact models to reason about their state conditions [112, 113], controlling the manipulator based on force/torque sensor readings [109, 120]. Model-based approaches using vision and

rigid systems have been difficult to implement, as uncertainty in the manipulator’s pose can create dangerously large forces [121]. Compliance has become a key component to overcoming this issue, both in software-based [108] and hardware-based [115,117] architectures. Many works have disregarded the use of a robot hand, which would be particularly required for pick-and-place tasks. Two previous works utilized a dexterous hand [122,152], where both found it advantageously extended the robot’s workspace. In both cases, *a priori* knowledge of the hole pose was available.

Learning-based solutions help deal with robot sensor and model uncertainty, and typically require physical environment exploration or some form of human-in-the-loop demonstrations [110]. Reinforcement learning and self-supervised learning [123,124] have been popular approaches, allowing the robot to interact with its environment and sufficiently explore desired regions [114,116,125,126]. Notably, learning manipulation policies is both, time consuming and data intensive, which increases the chance of damaging the robot.

While attempting to maintain a model-free nature but also limiting the need for expensive exploration, we are interested in forming a solution that can achieve reliable insertions of tight tolerance by solely using force data and vision feedback, and without *a priori* knowledge of the hole pose.

## Contact Formations

*Contact formations* ”provide a qualitative description of how 2 or more objects make contact with one another (e.g., vertex to surface, edge to edge)” [149]. This formulation is advantageous, in that it implicitly defines the contacts and thus constraints imposed on an object’s motion. Contact formations have been computed in different ways: from using CAD models [153] to probabilistic frameworks based on interaction [154]. From knowing these contact formations, other works have utilized them in planning [155] and control frameworks [156]. Fundamentally, the ability to group together different combinations of contacts that define the same or mere similar constraints, can be a powerful tool for defining more capable robot manipulation capabilities.

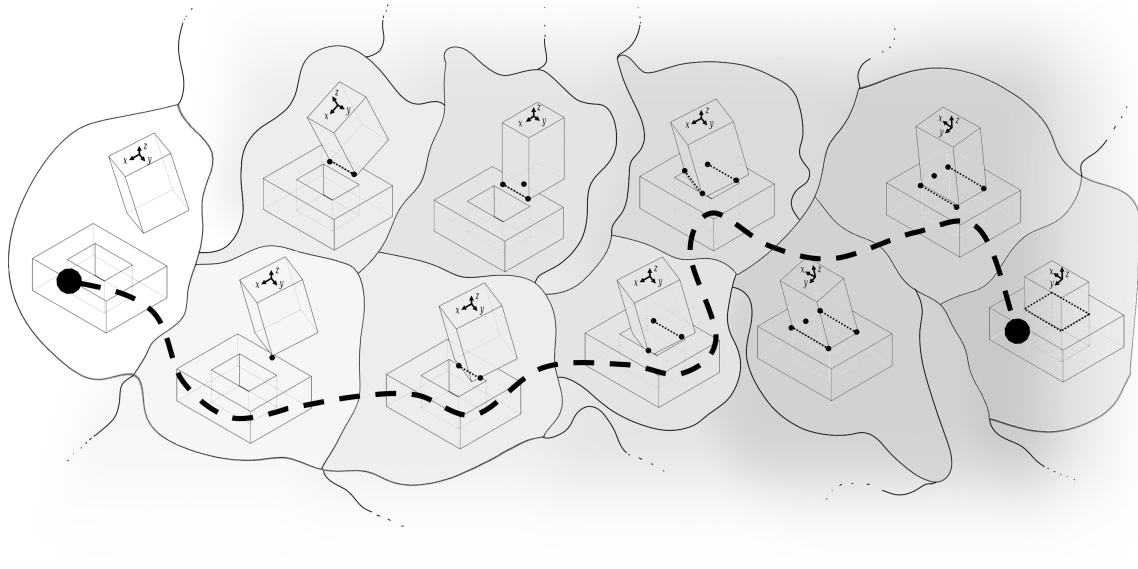


Figure 5.10: The Contact Formation-space (CF-space) of an insertion task can be conceptualized as an explicit organization of different contact formations that share borders according to constraint similarity and possible transitions. Transitions between contact formations are represented when constraints are added or removed to the state of an object. By controlling paths through a CF-space from light (few constraints) to dark (more constraints), the object follows a progression towards insertion. **Note:** This depiction of contact types is not exhaustive and other intermediate formations may be possible.

### 5.3.3 Methodology

We are interested in solving a generalized peg-in-hole problem by leveraging *compliance-enabled contact formations*—the concept that object constraints can be more easily acquired, broken, and remade when operating within a compliant robot’s “reconfiguration range” [157]. Conceptually, compliance allows the robot to convert a traditionally difficult force control problem into a position/velocity control problem, as there now exists an ability to “take up the slack” in control uncertainty. A F/T sensor is leveraged to modulate forces closed-loop. This is particularly advantageous for maintaining a contact formation, as now we can more easily ensure the system maintains constraints while operating along a desired path.

Fundamentally, contact formations (CF) represent groupings of discrete positional and physical relationships. Solving the insertion problem can thus be framed as a traversal from a starting,  $CF_0$ , toward a final,  $CF_n$ , transitioning through intermediate CFs, i.e.,  $CF_{path} = \{CF_0, CF_1, \dots, CF_n\}$  (Fig. 5.10). In this work, we demonstrate the efficiency of

this approach by following a fixed path in CF-space.

Formally, transitioning from one CF,  $CF_n$ , to a new CF,  $CF_{n+1}$ , along a trajectory in CF-space is comparable to adding or removing constraints on the object’s available DOF. Thus, the role of compliance serves to ensure that a single desired CF is maintained while manipulation is occurring via force modulation, and transitions are detected when *the motion of the object is impeded by the environment*. Once a variational change in F/T signals are felt by the robot, a contact formation change is detected, and the system has transitioned in CF-space. Conceptually, *move along a single axis until no longer possible, and while maintaining that force, move orthogonally until another constraint is added*.

The theory of controlling compliance-enabled contact formations is that constraints can be continually added until the task of insertion is complete. Notably, any DOF that is not currently constrained can be controlled to perform other tasks without modifying the current CF of the system. For example, an edge contact between an object and the hole’s surface allows the robot to explore via sliding until a lateral force is detected and thus a new CF is achieved. It is important to note that under this additive force contact process, explicitly estimating the current CF is possible if the starting CF is known and the transitions can be detected via F/T signals.

### **Assumptions and Prior Knowledge**

By relying on F/T sensing for environment perception, our algorithm leverages the following priors:

- The object starts in a stable, centered, and upright grasp.
- $CF_0$  has 0 constraints, i.e., 6 free DOFs.
- The hole is somewhere within a known 2D workspace boundary (x/y).
- Minimal knowledge of the object’s face type is known, either circular or non-circular.
- There exist a hole with positive tolerance matching the grasped peg within the workspace.
- There exists a low contact friction interaction between the object and the hole.



### CF-space Insertion Algorithm

Our approach follows a desired path in CF-space, as defined through a total of 7 steps. For ease of notation, we set the object frame as an orthogonal frame with the  $x$  – *axis* pointing toward the direction of the object motion on the hole surface plane, and the  $z$  – *axis* pointing downward. For each step, we provide a corresponding implicit CF control target. Although we define a set of values for force modulation, this is for clarity and is in practice robot-specific (see Sec. 5.3.4).

Let’s assume we can detect the object forces,  $F = \{f_x, f_y, f_z\}$  and torques,  $T = \{\tau_x, \tau_y, \tau_z\}$ , during insertion. Our goal is for the robot to traverse through a desired trajectory of CFs  $\{CF_0, \dots, CF_5\}$  (Fig. 5.9). We outline the algorithm in pseudocode below, and provide a breakdown on a real robotic system in Sec. 5.3.4.

---

## Insertion Pseudocode

---

### 1. Reaching the hole plane

*Current constraints:* None

*Target constraint:*  $(+f_z, 1.5\text{N})$

*Implicit CF targets:*  $(CF_1)$  point or edge on face

*Additional motions:*

- Lateral exploration within the workspace area, randomly selecting x/y direction within workspace limits.
- In-hand manipulation to tilt the object toward the direction of the lateral motion, to ensure an edge/face contact, and avoid a face/face contact.

### 2. Searching for the hole

*Current constraints:*  $(+f_z, 1.5\text{N})$

*Target constraint:*  $(+f_x, 0.7\text{N})$

*Implicit CF target:*  $(CF_2)$  3-point contact with hole

*Additional motions:* Lateral exploration as previous step.

### 3. Wedging

*Current constraints:*  $(+f_z, 1.5\text{N}), (+f_x, 0.7\text{N})$

*Target constraint:*  $(+f_y, 0.7\text{N})$

*Implicit CF target:*  $(CF_3)$  4-point contact with hole

### 4. Rotational alignment of peg and hole

*Current constraints:*  $(+f_z, 1.5\text{N}), (+f_x, 0.7\text{N}), (+f_y, 0.7\text{N})$

*Target constraint:*  $(+\tau_z, 0.1\text{N/m})$

*Implicit CF target:*  $(CF_4)$  hinge-type contact

*Note:* This step applies only to non-cylindrical objects.

### 5. Correcting upward tilt

*Current constraints:*  $(+f_z, 1.5\text{N})$

*Target constraint:*  $(+f_x, 0N), (+f_y, 0N)$

*Implicit CF target:*  $(CF_5)$  Prismatic joint-type contact

*Additional motion:* Rotation around  $x$ - and  $y$ -axes to minimize the accumulated angle between the object and the hand due to fingertip slip during the previous steps. This rotation is performed both with in-hand manipulation and arm motions.

*Note:* The lateral forces are now minimized to avoid jamming the object. This also helps centering the object in the hole if the peg rotation is not perfectly centered on the object’s center of mass. The angle information is provided by extracting the 3D pose of a marker placed on the surface of the object as seen from the palm camera.

## 6. Inserting peg

*Current constraints:*  $(+f_z, 1.5N), (+f_x, 0N), (+f_y, 0N)$

*Exit condition:* When the fingertips start touching the hole surface or the object hits the hole bottom, we switch to disengagement. This can be detected as a sharp increase in  $f_z$ .

*Note:* We have already reached the CF state that enables peg insertion, thus the system maintains it while the final free DOF, i.e., the  $z$ -axis translation, is controlled to perform the final insertion motion.

## 7. Detaching hand grip and retracting arm

*Action:* The hand opens and the arm returns to its origin position in an open loop motion.

---

### 5.3.4 Experiments

We evaluate our algorithm on a low-impedance manipulator and a compliant, underactuated hand (Fig. 5.11). The manipulator, a 7-DOF Barrett WAM, utilizes the *RRTConnect* algorithm via OMPL [145] for global planning and is controlled locally via a Jacobian-based velocity controller. The manipulator is imprecise due to an inaccurate internal model of its true system dynamics, which further challenges our algorithm’s robustness. The end effector, an adapted Yale OpenHand Model O [49], is a passively adaptive hand consisting of three actuators and six total joints. The hand is not equipped with tactile sensors or joint

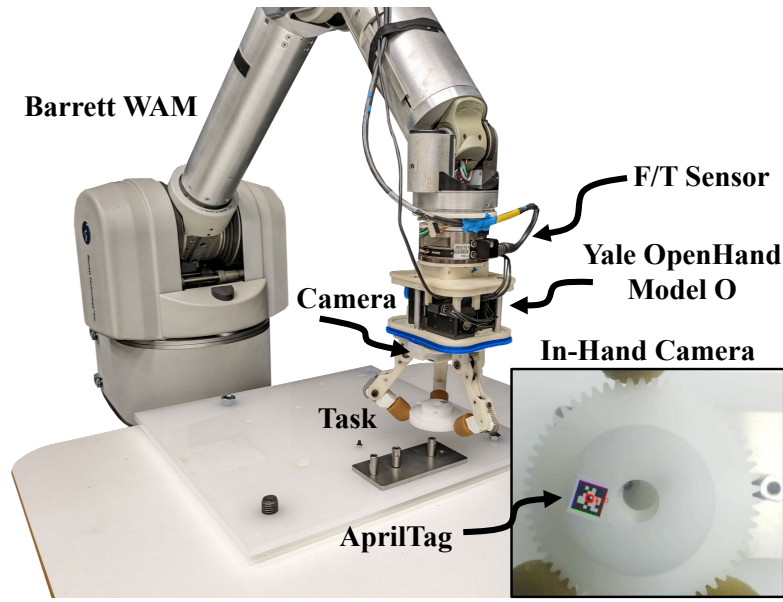


Figure 5.11: A Yale OpenHand Model O and a 6-axis force/torque sensor are affixed to the end of Barrett WAM manipulator. Inside of the palm of the hand, an in-hand camera setup is fabricated as to monitor the state of the object during manipulation via an AprilTag.

encoders, as to reduce weight and cost, making estimation of the true system state difficult during manipulation [100]. Modifications to the readily available open source design include rounded fingertips, as to facilitate in-hand manipulation, and bearings within the joints. An in-hand manipulation controller is devised, as in [106], and is utilized for fine motor control of object orientation up to  $\pm 20^\circ$ . Connecting the hand to the arm is a 6-axis ATI force/torque sensor sampled at 30Hz. Finally, a camera is fabricated into the palm of the hand as to enable the monitoring of object poses during manipulation via AprilTags [51].

The algorithm is validated through a variety of experiments. In our first experiment, we develop a linear pusher, comprised of a leadscrew and a stepper motor, to displace objects along different axes of the gripper's workspace. During this test, we measure the forces exerted on the object by the pusher and note the amount of force the gripper can resist before fingertip slip occurs. Thereafter, we test our algorithm with tight tolerance ( $< 0.25\text{mm}$ ) objects, objects commercially available in a children's insertion toy, and two tasks within the NIST Assembly Task Board #1 [158] (Fig. 5.12).

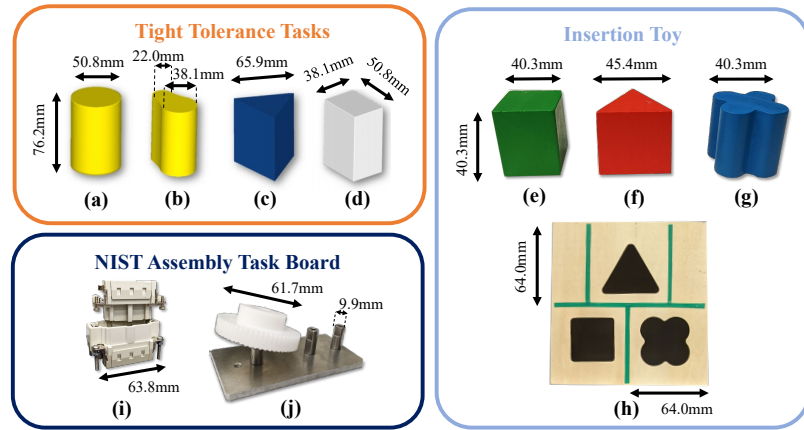


Figure 5.12: Tested objects can be classified into three categories: Tight Tolerance Tasks, Insertion Toy, and NIST Assembly Task Board. Objects are referenced according to their face geometries: (a) circle, (b) pear, (c) large triangle, (d) rectangle, (e) cube, (f) small triangle, and (g) clove. Subfigure (h) illustrates the insertion toy’s hole layout with designated search spaces in green. NIST objects (i) and (j) are referred to as the plug and gear, respectively.

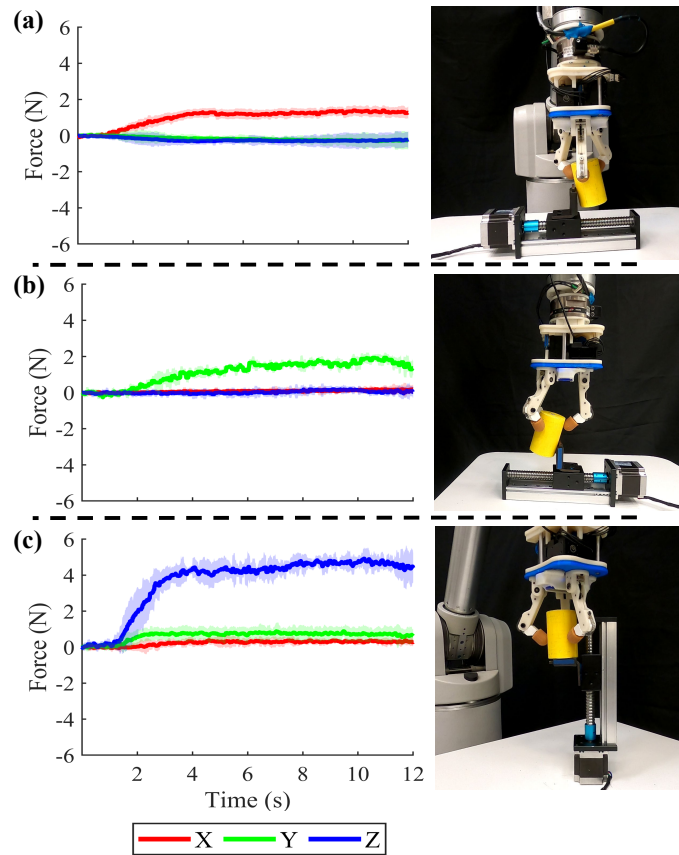


Figure 5.13: An object is pushed along the (a) $x$ -, (b) $y$ -, and (c) $z$ -axes with a linear pusher to evaluate the *force plateau* along each dimension, or more specifically, the amount of force the hand can resist before slip occurs.

## Quantifying Axial Compliance

First, we are interested in quantifying the compliance of our system. Conceptually, a purely compliant system with fixed contact could comply indefinitely until a hard stop is reached, but practically for grippers, there exists a limit in which the forces applied to an object can be resisted by fingertip contacts. This information is valuable to quantify as it enables the system to predict when slippage may occur, which can in turn be used to inform contact formation switching.

Given the linear pusher operating at a velocity of 3mm/s, we push the **(a)** circle grasped by the manipulator along the  $x$ -,  $y$ -, and  $z$ -axes for a duration of 12 seconds and over six trials. After each trial, the system is systematically reset. We record the forces measured by the F/T sensor via the pusher during each trial, and plot the mean and standard deviation for each evaluation (Fig. 5.13). Note that for each linear push, a *force plateau* is distinguishable, at  $\sim 1.5\text{N}$  for the  $x$ - and  $y$ -axes (Fig. 5.13(a)(b)) and at  $\sim 4\text{N}$  for the  $z$ -axis (Fig. 5.13(c)). These plateaus correspond to the hand's ability to resist forces in corresponding directions, or more physically, the point at which static friction of the contact is overcome and a new hand-object configuration is realized.

This data is valuable in that we are able to quantify the stable contact operating region given an external force. Once a force level is exceeded, sliding occurs and the hand-object state reconfigures until stability is once again realized. We can use these force plateaus for defining modulating forces for our algorithm in Sec. 5.3.4.

## Tight Tolerance Insertion

**Rectangle Case Study:** With knowledge of the *force plateaus*, we utilize this information to acquire and maintain contact formations. Leveraging our algorithm (Sec. 5.3.3), we set target modulating forces to 0.7N, 0.7N, and 1.5N along the  $x$ -,  $y$ -, and  $z$ -axes, respectively, which is  $\sim 40\text{-}50\%$  of the maximum force before slippage occurs.

In our first experiment, we attempt to insert the tight tolerance **(d)** rectangle with an initial  $27.1^\circ$  axial offset along the object's  $z$ -axis (Fig. 5.14(a)). Notably, the exact location of the hole is unknown, and the system is given a search space of  $8\text{cm} \times 8\text{cm}$ . The

Table 5.3: METRICS FOR OBJECT INSERTION EXPERIMENTS

	<b>Obj.</b>	<b>Tol. (mm)</b>	<b>Explore (s)</b>	<b>Insert (s)</b>	<b>Offset (°)</b>
<b>(a)</b>	circle	0.25	37.2	28.0	*n/a
<b>(b)</b>	pear	0.25	31.7	94.7	41.1
<b>(c)</b>	l. triangle	0.25	34.1	65.8	29.8
<b>(d)</b>	rectangle	0.25	44.2	48.1	27.1
<b>(e)</b>	cube	3.0	27.3	30.8	29.2
<b>(f)</b>	s. triangle	2.1	45.1	21.2	25.0
<b>(g)</b>	clove	2.6	63.2	19.4	23.7
<b>(i)</b>	plug	0.9	29.8	31.0	36.2
<b>(j)</b>	gear	0.1	27.9	21.4	*n/a

\*not applicable for pegs with circular faces

process begins by first finding a downward force, i.e., the first constraint, and choosing an exploration direction (in this case  $+f_x$ ) within the search space until an additional constraint is detected (transition 3). The additional force spike on the  $x - axis$  signals the hole’s perimeter and a force transition begins. This process is evidenced in Fig. 5.14(b), where forces are continually added and modulated around setpoints for states 1-7 (Sec. 5.3.3). Note that although compliant, the WAM robot has difficulty modulating velocities, and thus forces, due to controllability. Moreover, forces in Fig. 5.14(b) are represented in the world frame to represent rotations during steps 4-5 and are smoothed for clarity, so force spikes are not represented.

**All Objects:** Generalization is underscored by our ability to insert objects of varying convex, or near-convex, geometries—specifically pegs with circle-, pear-, and triangle-shaped faces (Fig. 5.12). The result of our evaluations with tolerances, exploration times, insertion times, and offset degrees is presented in Table 5.3(a-d). During evaluation, we noted that the **(a)** circle was the easiest, as it did not require  $z - axis$  offset control. The other objects **(b-d)**, were more difficult and posed various challenges. First, the initial offset was different for each, ranging from  $27.1^\circ$  to  $41.1^\circ$ . Objects **(c)** and **(d)** had sharp edges which encouraged jamming, whereas object **(b)** presented difficulty due to its non-convexity. Overcoming these challenges, we were able to complete insertions successfully with objects of  $<0.25\text{mm}$  tolerance and without prior knowledge of the hole pose (Fig. 5.14(c)).

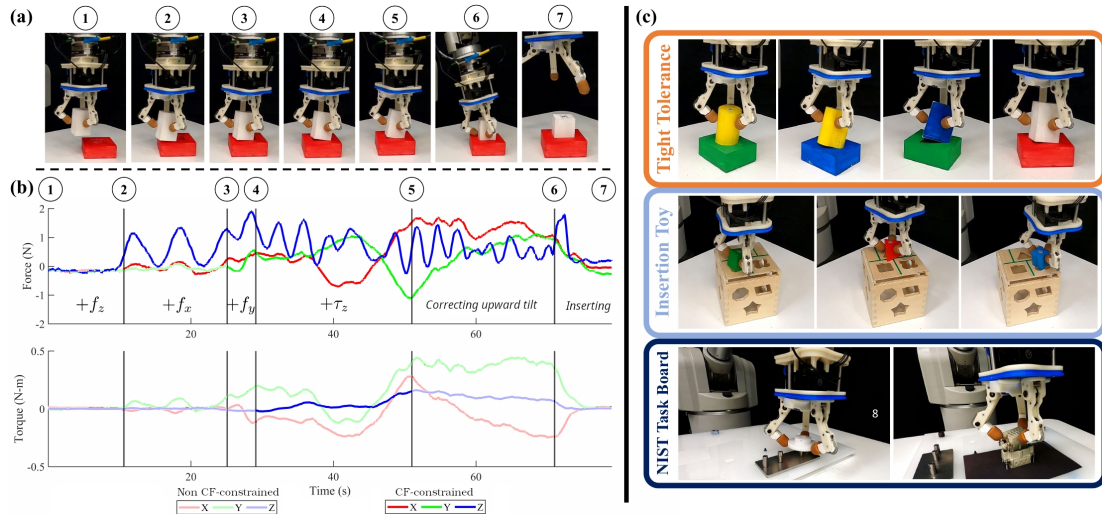


Figure 5.14: The progression of insertion is depicted in (a), where an object goes from free space (0 constraints) to inserted (5 constraints) into its goal configuration. (b) During this process, forces are modulated and added through different steps in the insertion task (forces are smoothed and placed in the world frame for clarity). (c) We evaluate this algorithm with tight tolerance tasks, insertion toys, and objects from the NIST Task Board.

### Open World Insertion Tasks

Beyond our tight tolerance evaluations, we were interested in applicability of our method to open world tasks. Our first experiment is with a commercially available children’s toy consisting of different object geometries and with a hole tolerance of approximately 2.5mm. Similar to the previous experiments, the pose of the hole is unknown, and the search area is now confined to a  $6.4\text{cm} \times 6.4\text{cm}$  space (denoted in green in Fig. 5.12(h)). Here, we attempt to challenge the exploration component of our algorithm, ensuring the object started at the edge of the search space. All insertions were successfully completed, with the longest exploration phase of 63.2 seconds for the (g) clove. Interestingly, the non-convexity of the clove did not complicate the insertion process as much as originally believed, as the round edges of the object helped limit jamming from occurring.

Our final experiment evaluated plug and gear insertion from the NIST Assembly Task Board (Fig. 5.14(c)). The gear task is interesting in that the peg-in-hole paradigm is transformed instead into a hole-on-peg schema. This was not a problem for our system, as the search pattern was instead completed on the bottom of the hole, i.e., the gear, instead of by using a peg. Similarly, the plug insertion with a relaxed plunger spring was



completed with ease, which started with an  $z$  – *axis* offset of  $36.2^\circ$  (Table 5.3). These tests underscore the practicality and generalizability of our method, which is further showcased in the supplementary video.

### 5.3.5 Discussions

This work presents a method that leverages *compliance-enabled contact formations* as a step towards tight tolerance insertion for robots. Our algorithm is simple, yet mechanically grounded, and exploits the concept of reducing uncertainty through additive contact constraints, i.e., manipulation funnels. Notably, in this work we do not need to utilize costly learning frameworks or system-specific, idealized analytical models—this method is effective yet did not require a single equation to describe in detail.

The authors are excited about this preliminary exploration, as our experiments illustrate validity of our approach for future applications. As an attempt to not overclaim contributions, the authors want to be forthcoming on known limitations that will warrant future investigation:

1. We cannot claim theoretical guarantees that an insertion will always be successful. In simulation, we verified that convex objects of non-negative tolerances should always succeed, but guarantees are not as clear for all non-convex circumstances.
2. Hole geometry requires a low-frictional “platform” for object exploration in order to find constraints. If this platform has variable friction, contact states may be transitioned prematurely and cause failure.
3. Negative tolerance insertion would be unlikely due to the maximum forces the system can apply (Sec. 5.3.4). A redesign of the end effector to apply a greater amount of force ( $\sim 5$ - $10$ N) would be beneficial for varied tasks.

Overall, the authors believe compliance will continue to prove invaluable for advancements in robot manipulation. By investigating how to build more capable end effectors and by developing robust control strategies for non-convex object insertion, our method should extend to a vast array of everyday insertion tasks for service robots of the future.

# Chapter 6

## Conclusion

### 6.1 Summary

In this thesis, I explored ways in which we can observe, control, and plan compliant robot manipulation online. Fundamentally, compliance benefits a system in its ability to passively adapt to the uncertainty a robot has with the world. This adaptive, and somewhat emergent, nature is beneficial in many cases a robot encounters, but also adds intricacies associated with planning and control. This realization leads us to the overarching theme of this thesis:

*~ Complex robots complicate control; keep designs simple and exploit emergent behavior ~*

I began my story by understanding ways in which we can observe the state of a hand-object system – developing a generalized approach as how to predict when various contact phenomena, such as sliding, drop, or stuck cases were likely to occur. I showed just how far my generalization went by training on one robot hand and testing on others. I also provided theoretical bounds by which my method should hold. We generally found that kinematics and grasp mechanics were enough to describe state, and these metrics can be extracted solely through visual tracking and without any other sensing.

As we learned more about ways to understand state, and that we could sufficiently define the general state of a hand-object system purely through kinematics, I investigated ways to control a grasped object. To challenge our approach, we took our planar problem and made it spatial, i.e., 3D, and developed an energy-based model to describe fingertip manipulation with an underactuated hand. As we found initially, our model became slightly inaccurate during online manipulation due to changing of kinematic parameters, e.g., effective link lengths due to rolling, and was also subject

to only fully constrained, point-to-point trajectories. To overcome these obstacles, we introduced our final method of controlling such hands through an online Model Predictive Control framework.

Thereafter, by being able to observe and control the state of a hand-object system, we were finally interested in in-hand planning for increasing the workspace of the robot. Succinctly, *finger-gaiting* is the act of freely making and breaking contact with an object during manipulation. As contacts are made, different motion manifolds are available to the robot. Planning between these manifolds, i.e., where and when the robot should make and break contact, becomes the focus of this work. Generally, we find that our bidirectional multi-modal planning approach is beneficial in that it is a fast, online approach that can recover from external perturbations, which is only possible because of the compliance of the system. Formally, we show how modal transfer configurations turn into modal transfer regions, underscoring the main benefit of planning with compliant robots.

To conclude this thesis, we wanted to expand our findings for in-hand manipulation to those of tight tolerance and open world assembly tasks. Empirically, we discussed and evaluated how compliance enables us to “take up the slack” in modeling uncertainties with our robot, and allows us to plan and control online whole-arm trajectories faster and more effectively. Fundamentally, our system was able to complete traditionally difficult or impossible insertion tasks with simple controllers and simple sensors, underscoring our continued narrative for the benefit of compliance for the future of robot manipulation.

## 6.2 Lessons Learned and Future Work

Though this work makes progress in realizing fingertip-based in-hand manipulation and generalized whole-arm assembly for robotics, there is much work still to be done. First and foremost, and very notably, the manipulation capabilities we have been able to provide on physical systems are still very elementary – these mainly consist of picking up simple and convex objects, *fidgiting* with it, and placing is as needed. In fact, I would likely characterize these capabilities as being somewhat similar to a 2-3 year-old human’s. The assumptions we imposed on our manipulation procedure in much of this work, e.g., fingertip-based manipulation, low object mass, are not ever so apparent in daily human manipulation. Thus, extending this study to whole-hand control, i.e., using proximal phalanges of the hand, would especially be necessary.

Beyond these higher-level critiques of the manipulation capabilities we were able to accomplish, there is still much work to be done on the theoretical formulation associated with compliance in manipulation. As we continued through this research and thought about compliance in terms of energy, we identified an interesting formalization that needs to be made – the concept of energy-

based constraints on object motion represented as funnels. Manipulation funnels in the literature as originally presented by Mason [22] had “hard” constraints, as they were mostly in terms of geometry/kinematics. Through our studies, we find that there exist “hard” and “soft” constrained regions in the form of a differentiable funnel. This true formalization begs much future work, e.g., mathematical properties of funnels, representation of funnels, and control through concatenated funnels. I look forward to investigating this idea more with colleagues in the future.

The lessons learned from this work were numerous, but I will attempt to highlight just a few. First and foremost, we have shown that high fidelity tactile sensing is not necessary for tasks that were once thought to be impossible without it. By closing the control loop through vision, we were able to accomplish an array of tasks without complicating hand design. Second, sometimes having control over everything really equates to having control over nothing. Throughout this work, we exploited the properties of compliant mechanisms, which were the sole reason we were able to make such progress with these tasks in the first place. As we were able to acquire and maintain a grasp without the already computational overhead of impedance/admittance/stiffness control solutions on fully actuated hands, this facilitated all research described in this thesis. And, finally, manipulation is a fundamentally difficult problem and is not going to be solved overnight – roboticists have been working on this for nearly half a century. In my view, it serves as one of the greatest challenges we will ever face in robotics. I don’t see what the golden ticket will be for this in the near term, but it is important that we continue to explore different avenues rather than conforming to a single thread of research. Throughout this journey, I have been fortunate to be one of the first researchers to work on compliant in-hand manipulation. To this end, I hope to keep reading more explorative papers in manipulation in the near future until we find that breakthrough.

### **6.3 Suggestions for Continuing this Line of Work**

Observing, controlling, and planning with mechanically compliant robots is a challenging endeavor. I do not say this solely because I worked on in-hand manipulation, which is a fundamentally difficult problem, but because these sorts of robots generally introduce many mental obstacles for the researcher. Normally in our academic endeavors, we are trained to want to know everything we can about our system – every sensor reading, every computational timestamp, and every actuation velocity. But with these systems, you need to get comfortable in dealing with the fact that much about the system is unknown. To this end, try to exploit this concept for the gain of the task. Here are some rules of thumb that helped me:

1. Get comfortable not knowing everything happening with your robot. Observe and control what you can. If something else needs controlled, add minimal complexity and try again.
2. Simulation continues to be poor at describing interactions with the real world. Compliance allows us to deal with intricacies of the real world. Train and test with this in mind.
3. No robot will act ideally. The number of times I tried to reduce friction in actuation transmissions was too high to count. Deal with this through other means.
4. Emergent, parasitic, compliant (whatever word you fancy for the day) motion will be how we solve manipulation. If this is mechanical-based or software-based is still up for discussion. Try to think why you are using compliance, how it helps, and if there are better ways to implement it. Each method has benefits. Try to figure out what may be yours.

Good luck in your endeavors. I hope that our paths will cross someday and that we will all continue to push towards general purpose robot manipulation for the future.

# Appendix A

## Notes on Visual-based Insertion

### Selecting a Manipulation Frame

Selecting the best manipulation frame for object insertion, as eluded to in Sec. 5.2.4, is important as the antipodal grasp that defines  $\mathcal{M}$ , limits the amount of torque that can be applied by the contacts onto the object. Let's consider two objects that define different antipodal contact widths,  $d_o^1$  and  $d_o^2$ , respectively. These distances fundamentally define the lever arm by which contact forces can be applied by the manipulator, or in the case of this work a hand, to the object. Assuming that friction coefficients and contact force applications are identical, the grasp with  $\max(d_o^1, d_o^2)$  will have a larger bound in the torques than can be imparted onto the object to aid in insertion.

In addition to the case of supplying a larger torque to the object, selecting the cross section of  $\Gamma$  that nearly maximizes  $d_o$  is also advantageous, according to the defined condition of insertion (Sec. 5.2.4). Fundamentally, while keeping the tolerance between the hole and all candidate object cross sections constant, as one increases  $d_o$ ,  $\beta_f$  will subsequently decrease. By selecting  $d_o$  such that it is (near) maximized, this condition is attempting to control the most difficult axis of the object to insert while relying on compliance to account for any out-of-plane reorientation of the other axis, which has a larger  $\beta_f$ . While the proposed approach for picking  $\mathcal{M}$  is largely heuristic, it suits well for insertion with a dexterous hand that can apply much smaller forces than that of a manipulator.

More formally, in the case of selecting  $\mathcal{M} \in SE(3)$ , we want to choose an insertion plane inside of an arbitrary 3D object geometry,  $\Gamma$ , such that we maximize the projection of  $T$  back onto the vector connecting the two antipodal contact points. Assuming a given  $\Gamma$ , we can compute  $T$  and thus  $\mathcal{M}$  by analyzing properties of the object's face. More formally, consider that the outer geometry of a peg's face is sufficiently represented by a 2D point cloud of  $n$  points, i.e.,  $F \in \mathbb{R}^{n \times 2}$ . It is possible to calculate the centroid of the point cloud,  $F_c$ , by taking its dimension-wise mean. Using Principal

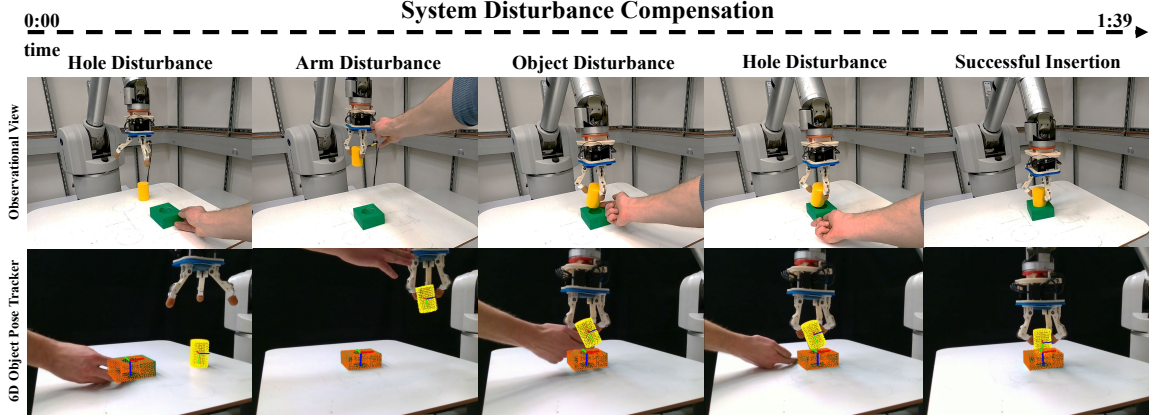


Figure A.1: The proposed system is robust to external disturbances imposed during task execution. In this task sequence, the pose of the hole, arm, and object are all manually perturbed and the system recovers such that the object successfully reaches its goal configuration. Refer to the supplementary video for the complete results.

Component Analysis (PCA), we can compute the two principal axes of  $F$ , forming  $P = [\pi_1, \pi_2]$ . We solve for the outermost point,  $m$ , of the point cloud geometry by solving the optimization problem,

$$m = \arg \max_{f \in F} \pi_1 \cdot (f - F_c) \quad (\text{A.1})$$

The location of  $m$ , which lies, or almost lies, along  $\pi_1$ , appropriately maximizes the distance,  $d_o$ , between two antipodal contacts on the 3D object geometry. We can then compute the transformation  $T$  from  $m$  back to the object frame,  $\mathcal{X} \in SE(3)$ , and thus providing our manipulation frame  $\mathcal{M} \in SE(3)$  that will be used for guiding object insertion.

### System Recovery to Disturbances

For robots that act in unknown, and contact-rich environment, various forms of disturbances can arise depending on the robot's operating scenario. As aforementioned, these disturbances can be self-induced, where for instance, a robot's perception and control noise causes the manipulated object to move differently than the internal model predicted, and thus the system must react. Similarly, disturbances can occur that are not caused directly by the robot's actions but by other objects or robots in its environment, as in the case of highly cluttered scenarios. We simulate such occurrences by deliberately disturbing the arm, object, and hole during an execution (Fig. A.1). More specifically, during the task and in four separate occasions, we move the location of the hole on the support surface, causing our system to reactively adapt to the new goal configuration via the continuous feedback from the object tracker. During this sequence, we also push on the arm

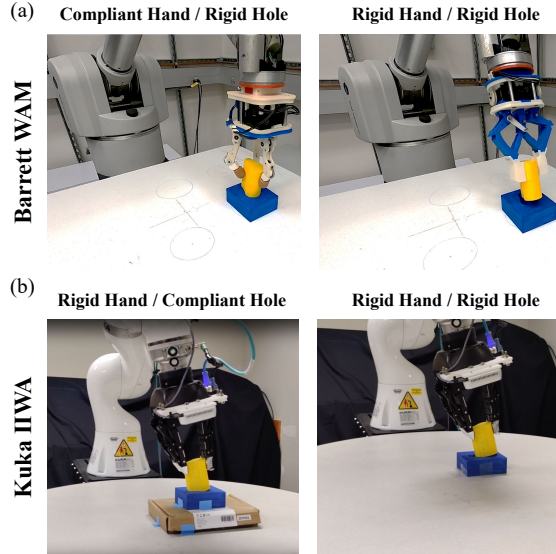


Figure A.2: The ablations make use of two manipulators with different levels of compliance at the arm, hand, or environment level. (a) A Barrett WAM serves as a low-impedance, compliant manipulator and has been tested with a compliant (left) and a rigid (right) hand; while the (b) Kuka IIWA is an example of a rigid manipulator, which has been tested with a rigid hand and a compliant (left) or rigid (right) hole.

and the object, perturbing the state of the hand-object configuration, as to require the system to overcome such disturbances for successful insertion. These task evaluations are included in this paper’s supplementary video.

### System Observations and Limitations

This subsection is part of a discussion on system limitations (Sec. 5.2.6), and what can be improved about the system for future advancements in robot assembly tasks.

**Visual Perception:** While the RGBD-based 6D pose tracker is robust to a variety of objects with challenging properties such as textureless, reflective, geometrical featureless, or thin/small shapes, it struggles to track severely shiny, glossy, or transparent objects, due to the degenerated depth sensing of the camera. In future work, we hope to explore extending this framework to these other types of scenarios with the techniques of depth enhancement and completion [159]. In addition, the current framework requires an object CAD model beforehand to perform 6D pose tracking and for reasoning about the task of peg-in-hole insertion. In future work, reconstructing the model of novel objects on-the-fly [136] while with sufficient precision to perform high-tolerance tight insertion tasks is of interest.

**Inaccuracy of the Low-Impedance Manipulator for Grasp Acquisition:** While the manipulator leveraged in this work was largely beneficial for task completion, it also introduced



difficulties in acquiring an initial grasp. Soft, compliant, and underactuated hands are well suited for grasping, but in cases of robot assembly where future within-hand manipulation is necessary (especially with 40+ finger actions as we saw with our tasks), acquiring a well-intended and stable grasp at the fingertips is necessary from the start. Such grasps would be especially possible with an accurate manipulator that is able to appropriately position the hand over the object for grasping. Though, for cases in this work where we saw task failure, it was largely due to starting with a poor initial fingertip grasp, which was directly attributed to the end effector's deviation from its commanded initial pose.

# Bibliography

- [1] A. Bicchi. Hands for dexterous manipulation and robust grasping: A difficult road toward simplicity. *IEEE Transactions on Robotics and Automation*, 16(6):652–662, 2000.
- [2] W. Ruotolo, R. Thomasson, J. Herrera, A. Gruebele, and M. Cutkosky. Distal hyperextension is handy: High range of motion in cluttered environments. *IEEE Robotics and Automation Letters*, 5(2):921–928, 2020.
- [3] C. H. Kim and J. Seo. Shallow-depth insertion: Peg in shallow hole through robotic in-hand manipulation. *IEEE Robotics and Automation Letters*, 4(2):383–390, 2019.
- [4] T. Bhattacharjee, G. Lee, H. Song, and S. S. Srinivasa. Towards Robotic Feeding: Role of Haptics in Fork-Based Food Manipulation. *IEEE Robotics and Automation Letters*, 4(2):1485–1492, 2019.
- [5] R. Johansson. How is grasping modified by somatosensory input. *Motor control: concepts and issues*, 14:331–335, 1991.
- [6] N. Sharma and M. Venkadesan. Finger stability in precision grips. *Proceedings of the National Academy of Sciences*, 119:e2122903119, 2022.
- [7] W. G. Bircher, A. S. Morgan, and A. M. Dollar. Complex manipulation with a simple robotic hand through contact breaking and caging. *Science Robotics*, 6(54), 2021.
- [8] R. R. Ma and A. M. Dollar. On dexterity and dexterous manipulation. In *2011 15th International Conference on Advanced Robotics (ICAR)*, pages 1–7. IEEE, 2011.
- [9] J. v. d. Berg, S. Miller, K. Goldberg, and P. Abbeel. Gravity-based robotic cloth folding. In *Algorithmic Foundations of Robotics IX*, pages 409–424. Springer, 2010.
- [10] R. Balasubramanian, J. T. Belter, and A. M. Dollar. External disturbances and coupling mechanisms in underactuated hands. In *International design engineering technical conferences and computers and information in engineering conference*, volume 44106, pages 175–184, 2010.

- [11] O. M. Andrychowicz, B. Baker, M. Chociej, R. Jozefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray, et al. Learning dexterous in-hand manipulation. *The International Journal of Robotics Research*, 39(1):3–20, 2020.
- [12] B. Calli, K. Srinivasan, A. Morgan, and A. M. Dollar. Learning modes of within-hand manipulation. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3145–3151. IEEE, 2018.
- [13] B. Calli, A. Kimmel, K. Hang, K. Bekris, and A. Dollar. Path planning for within-hand manipulation over learned representations of safe states. In *International Symposium on Experimental Robotics*, pages 437–447. Springer, 2020.
- [14] A. S. Morgan, W. G. Bircher, B. Calli, and A. M. Dollar. Learning from transferable mechanics models: Generalizable online mode detection in underactuated dexterous manipulation. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 5823–5829. IEEE, 2019.
- [15] K. M. Lynch. Estimating the friction parameters of pushed objects. In *Proceedings of 1993 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'93)*, volume 1, pages 186–193. IEEE, 1993.
- [16] T. Okada. Computer control of multijointed finger system for precise object-handling. *IEEE Transactions on Systems, Man, and Cybernetics*, 12(3):289–299, 1982.
- [17] J. Kerr and B. Roth. Analysis of multifingered hands. *The International Journal of Robotics Research*, 4(4):3–17, 1986.
- [18] D. L. Brock. Enhancing the dexterity of a robot hand using controlled slip. In *Proceedings 1988 IEEE International Conference on Robotics and Automation (ICRA)*, pages 249–251.
- [19] A. M. Dollar and R. D. Howe. The Highly Adaptive SDM Hand: Design and Performance Evaluation. *The International Journal of Robotics Research*, 29(5):585–597, 2010.
- [20] J. C. Trinkle. A quasi-static analysis of dextrous manipulation with sliding and rolling contacts. In *ICRA*, pages 788–793, 1989.
- [21] R. M. Murray, S. S. Sastry, and L. Zexiang. *A Mathematical Introduction to Robotic Manipulation*. CRC Press, Inc., USA, 1994.
- [22] M. Mason. The mechanics of manipulation. In *Proceedings. 1985 IEEE International Conference on Robotics and Automation*, volume 2, pages 544–548. IEEE, 1985.

- [23] A. M. Okamura, N. Smaby, and M. R. Cutkosky. An overview of dexterous manipulation. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, volume 1, pages 255–262. IEEE, 2000.
- [24] K. Hertkorn, M. A. Roa, and C. Borst. Planning in-hand object manipulation with multifingered hands considering task constraints. In *2013 IEEE International Conference on Robotics and Automation*, pages 617–624. IEEE, 2013.
- [25] R. Michalec and A. Micaelli. Stiffness modeling for multi-fingered grasping with rolling contacts. In *2010 10th IEEE-RAS International Conference on Humanoid Robots*, pages 601–608. IEEE, 2010.
- [26] L. U. Odhner and A. M. Dollar. Dexterous manipulation with underactuated elastic hands. In *Proceedings 2011 IEEE International Conference on Robotics and Automation*, pages 5254–5260.
- [27] Y. Li and I. Kao. A review of modeling of soft-contact fingers and stiffness control for dextrous manipulation in robotics. In *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No. 01CH37164)*, volume 3, pages 3055–3060. IEEE, 2001.
- [28] N. Hogan. Impedance Control: An Approach to Manipulation. In *1984 American Control Conference*, pages 304–313, 1984.
- [29] D. Prattichizzo, M. Malvezzi, M. Gabbicini, and A. Bicchi. On motion and force controllability of precision grasps with hands actuated by soft synergies. *IEEE transactions on robotics*, 29(6):1440–1456, 2013.
- [30] R. Balasubramanian, J. T. Belter, and A. M. Dollar. Disturbance response of two-link underactuated serial-link chains. 2012.
- [31] C. Della Santina, C. Piazza, G. Grioli, M. G. Catalano, and A. Bicchi. Toward dexterous manipulation with augmented adaptive synergies: The pisa/iit softhand 2. *IEEE Transactions on Robotics*, 34(5):1141–1156, 2018.
- [32] A. Rocchi, B. Ames, Z. Li, and K. Hauser. Stable simulation of underactuated compliant hands. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4938–4944. IEEE, 2016.
- [33] T. Laliberté and C. M. Gosselin. Simulation and design of underactuated mechanical hands. *Mechanism and machine theory*, 33(1-2):39–57, 1998.

- [34] B. Calli and A. M. Dollar. Robust Precision Manipulation With Simple Process Models Using Visual Servoing Techniques With Disturbance Rejection. *IEEE Transactions on Automation Science and Engineering*, 16(1):406–419, 2019.
- [35] 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 Robotics and Automation Letters*, 4(2):1287–1294, 2019.
- [36] M. Kalakrishnan, L. Righetti, P. Pastor, and S. Schaal. Learning force control policies for compliant manipulation. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4639–4644. IEEE, 2011.
- [37] Y. Yang, Y. Li, C. Fermuller, and Y. Aloimonos. Robot learning manipulation action plans by” watching” unconstrained videos from the world wide web. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 29, 2015.
- [38] A. Gupta, C. Eppner, S. Levine, and P. Abbeel. Learning dexterous manipulation for a soft robotic hand from human demonstrations. In *Proceedings 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3786–3793.
- [39] N. Abdo, H. Kretschmar, L. Spinello, and C. Stachniss. Learning manipulation actions from a few demonstrations. In *2013 IEEE International Conference on Robotics and Automation*, pages 1268–1275. IEEE, 2013.
- [40] Z. Su, K. Hausman, Y. Chebotar, A. Molchanov, G. E. Loeb, G. S. Sukhatme, and S. Schaal. Force estimation and slip detection/classification for grip control using a biomimetic tactile sensor. In *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*, pages 297–303. IEEE, 2015.
- [41] M. R. Tremblay and M. R. Cutkosky. Estimating friction using incipient slip sensing during a manipulation task. In *[1993] Proceedings IEEE International Conference on Robotics and Automation*, pages 429–434. IEEE, 1993.
- [42] H. Dang and P. K. Allen. Learning grasp stability. In *2012 IEEE International Conference on Robotics and Automation*, pages 2392–2397. IEEE, 2012.
- [43] K. Hang, M. Li, J. A. Stork, Y. Bekiroglu, F. T. Pokorny, A. Billard, and D. Kragic. Hierarchical fingertip space: A unified framework for grasp planning and in-hand grasp adaptation. *IEEE Transactions on robotics*, 32(4):960–972, 2016.
- [44] Grasp quality measures: review and performance. *Autonomous Robots*, 38(1):65–88, Jan 2014.

- [45] T. Yoshikawa. Manipulability of robotic mechanisms. *The international journal of Robotics Research*, 4(2):3–9, 1985.
- [46] N. Vahrenkamp, T. Asfour, G. Metta, G. Sandini, and R. Dillmann. Manipulability analysis. In *2012 12th IEEE-RAS International Conference on Humanoid Robots (Humanoids 2012)*, pages 568–573. IEEE, 2012.
- [47] D. J. Montana. Contact Stability for Two-Fingered Grasps. *IEEE Transactions on Robotics and Automation*, 8(4):421–430, 1992.
- [48] W. G. Birchler, A. M. Dollar, and N. Rojas. A two-fingered robot gripper with large object reorientation range. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3453–3460, 2017.
- [49] R. Ma and A. Dollar. Yale openhand project: Optimizing open-source hand designs for ease of fabrication and adoption. *IEEE Robotics & Automation Magazine*, 24(1):32–40, 2017.
- [50] B. Calli and A. M. Dollar. Vision-based model predictive control for within-hand precision manipulation with underactuated grippers. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2839–2845. IEEE, 2017.
- [51] E. Olson. Apriltag: A robust and flexible visual fiducial system. In *2011 IEEE international conference on robotics and automation*, pages 3400–3407. IEEE, 2011.
- [52] A. Cole, J. Hauser, and S. Sastry. Kinematics and control of multifingered hands with rolling contact. In *Proceedings. 1988 IEEE International Conference on Robotics and Automation*, pages 228–233. IEEE, 1988.
- [53] J. L. Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975.
- [54] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.
- [55] J. C. Trinkle, J.-S. Pang, S. Sudarsky, and G. Lo. On dynamic multi-rigid-body contact problems with coulomb friction. *Journal of Applied Mathematics and Mechanics*, 77(4):267–279, 1997.
- [56] T. Okada. Computer Control of Multijointed Finger System for Precise Object-Handling. *IEEE Transactions on Systems, Man, and Cybernetics*, 12(3):289–299, 1982.

- [57] A. M. Dollar and R. D. Howe. The Highly Adaptive SDM Hand: Design and Performance Evaluation. *The International Journal of Robotics Research*, 29(5):585–597, apr 2010.
- [58] N. Fazeli, S. Zapolsky, E. Drumwright, and A. Rodriguez. Learning data-efficient rigid-body contact models: Case study of planar impact. *arXiv preprint arXiv:1710.05947*, 2017.
- [59] A. Rajeswaran, V. Kumar, A. Gupta, G. Vezzani, J. Schulman, E. Todorov, and S. Levine. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. *arXiv preprint arXiv:1709.10087*, 2017.
- [60] M. Andrychowicz, B. Baker, M. Chociej, R. Józefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray, J. Schneider, S. Sidor, J. Tobin, P. Welinder, L. Weng, and W. Zaremba. Learning dexterous in-hand manipulation. *The International Journal of Robotics Research*, 39(1):3–20, 2020.
- [61] A. S. Morgan, K. Hang, W. G. Bircher, and A. M. Dollar. A data-driven framework for learning dexterous manipulation of unknown objects. In *Proceedings 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 8273–8280.
- [62] 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 Robotics and Automation Letters*, 4(2):1287–1294, apr 2019.
- [63] H. van Hoof, T. Hermans, G. Neumann, and J. Peters. Learning robot in-hand manipulation with tactile features. In *Proceedings 2015 IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pages 121–127.
- [64] B. Sundaralingam and T. Hermans. Relaxed-rigidity constraints: kinematic trajectory optimization and collision avoidance for in-grasp manipulation. *Autonomous Robots*, 43(2):469–483, 2019.
- [65] B. Ward-Cherrier, N. Rojas, and N. F. Lepora. Model-free precise in-hand manipulation with a 3d-printed tactile gripper. *IEEE Robotics and Automation Letters*, 2(4):2056–2063, 2017.
- [66] B. Calli and A. M. Dollar. Robust precision manipulation with simple process models using visual servoing techniques with disturbance rejection. *IEEE Transactions on Automation Science and Engineering*, 16(1):406–419, 2018.
- [67] N. Furukawa, A. Namiki, S. Taku, and M. Ishikawa. Dynamic regrasping using a high-speed multifingered hand and a high-speed vision system. In *Proceedings 2006 IEEE International Conference on Robotics and Automation (ICRA)*, pages 181–187.

- [68] K. Hang, M. Li, J. A. Stork, Y. Bekiroglu, F. T. Pokorny, A. Billard, and D. Kragic. Hierarchical fingertip space: A unified framework for grasp planning and in-hand grasp adaptation. *IEEE Transactions on Robotics*, 32(4):960–972, Aug 2016.
- [69] K. Hang, W. G. Bircher, A. S. Morgan, and A. M. Dollar. Hand–object configuration estimation using particle filters for dexterous in-hand manipulation. *The International Journal of Robotics Research*, 2019.
- [70] K. Tahara, S. Arimoto, and M. Yoshida. Dynamic object manipulation using a virtual frame by a triple soft-fingered robotic hand. In *Proceedings 2010 IEEE International Conference on Robotics and Automation (ICRA)*.
- [71] B. Calli, A. Singh, J. Bruce, A. Walsman, K. Konolige, S. Srinivasa, P. Abbeel, and A. M. Dollar. Yale-cmu-berkeley dataset for robotic manipulation research. *The International Journal of Robotics Research*, 36(3):261–268, 2017.
- [72] L. Han and J. C. Trinkle. Dexterous manipulation by rolling and finger gaing. In *Proceedings. 1998 IEEE International Conference on Robotics and Automation (Cat. No. 98CH36146)*, volume 1, pages 730–735. IEEE, 1998.
- [73] B. Sundaralingam and T. Hermans. Geometric in-hand regrasp planning: Alternating optimization of finger gaits and in-grasp manipulation. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 231–238. IEEE, 2018.
- [74] G. Khandate, M. Haas-Heger, and M. Ciocarlie. On the feasibility of learning finger-gaiting in-hand manipulation with intrinsic sensing, 2021.
- [75] A. Bhatt, A. Sieler, S. Puhmann, and O. Brock. Surprisingly robust in-hand manipulation: An empirical study-supplementary material. 2021.
- [76] S. Abondance, C. B. Teeple, and R. J. Wood. A dexterous soft robotic hand for delicate in-hand manipulation. *IEEE Robotics and Automation Letters*, 5(4):5502–5509, 2020.
- [77] B. Wen, C. Mitash, B. Ren, and K. E. Bekris. se (3)-tracknet: Data-driven 6d pose tracking by calibrating image residuals in synthetic domains. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 10367–10373. IEEE, 2020.
- [78] B. Wen, C. Mitash, S. Soorian, A. Kimmel, A. Sintov, and K. E. Bekris. Robust, occlusion-aware pose estimation for objects grasped by adaptive hands. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6210–6217. IEEE, 2020.



- [79] R. R. Ma and A. M. Dollar. An underactuated hand for efficient finger-gaiting-based dexterous manipulation. In *2014 IEEE International Conference on Robotics and Biomimetics (ROBIO 2014)*, pages 2214–2219. IEEE, 2014.
- [80] K. Hang, W. G. Bircher, A. S. Morgan, and A. M. Dollar. Manipulation for self-identification, and self-identification for better manipulation. *Science Robotics*, 6(54), 2021.
- [81] K. Hang, A. S. Morgan, and A. M. Dollar. Pre-grasp sliding manipulation of thin objects using soft, compliant, or underactuated hands. *IEEE Robotics and Automation Letters*, 4(2):662–669, 2019.
- [82] A. Morgan, B. Wen, J. Liang, A. Boularias, A. Dollar, and K. Bekris. Vision-driven compliant manipulation for reliable; high-precision assembly tasks. *Robotics: Science and Systems XVII*, Jul 2021.
- [83] R. H. N. Chavan-Dafle and A. Rodriguez. Planar in-hand manipulation via motion cones. *IJRR*, 2019.
- [84] S. Yuan, L. Shao, C. L. Yako, A. Gruebele, and J. K. Salisbury. Design and control of roller grasper v2 for in-hand manipulation. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 9151–9158, 2020.
- [85] K. Hauser and J.-C. Latombe. Multi-modal motion planning in non-expansive spaces. *The International Journal of Robotics Research*, 29(7):897–915, 2010.
- [86] M. Zucker, N. Ratliff, A. D. Dragan, M. Pivtoraiko, M. Klingensmith, C. M. Dellin, J. A. Bagnell, and S. S. Srinivasa. Chomp: Covariant hamiltonian optimization for motion planning. *The International Journal of Robotics Research*, 32(9-10):1164–1193, 2013.
- [87] Z. Kingston, A. M. Wells, M. Moll, and L. E. Kavraki. Informing multi-modal planning with synergistic discrete leads. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3199–3205. IEEE, 2020.
- [88] J. J. Craig. *Introduction to robotics: mechanics and control, 3/E*. Pearson Education India, 2009.
- [89] L. Euler. Formulae generales pro translatione quacunque corporum rigidorum. *Novi Commentarii academiae scientiarum Petropolitanae*, pages 189–207, 1776.
- [90] D. Huynh. Metrics for 3d rotations: Comparison and analysis. *Journal of Mathematical Imaging and Vision*, 35:155–164, 10 2009.

- [91] A. S. Morgan, K. Hang, and A. M. Dollar. Object-agnostic dexterous manipulation of partially constrained trajectories. *IEEE Robotics and Automation Letters*, 5(4):5494–5501, 2020.
- [92] S. Cruciani, B. Sundaralingam, K. Hang, V. Kumar, T. Hermans, and D. Kragic. Benchmarking in-hand manipulation. *IEEE Robotics and Automation Letters*, 5(2):588–595, 2020.
- [93] B. Wen and K. Bekris. Bundletrack: 6d pose tracking for novel objects without instance or category-level 3d models. *IROS*, 2021.
- [94] C. C. Kemp, A. Edsinger, and E. Torres-Jara. Challenges for robot manipulation in human environments [Grand Challenges of Robotics]. *IEEE Robotics & Automation Magazine*, 14(1):20–29, 2007.
- [95] O. Kroemer, S. Niekum, and G. D. Konidaris. A Review of Robot Learning for Manipulation: Challenges, Representations, and Algorithms. *CoRR*, abs/1907.03146, 2019.
- [96] N. Correll, K. E. Bekris, D. Berenson, O. Brock, A. Causo, K. Hauser, K. Okada, A. Rodriguez, J. M. Romano, and P. R. Wurman. Analysis and Observations From the First Amazon Picking Challenge. *IEEE Transactions on Automation Science and Engineering*, 15(1):172–188, 2018.
- [97] J. van den Berg, S. Miller, K. Goldberg, and P. Abbeel. *Gravity-Based Robotic Cloth Folding*, pages 409–424. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.
- [98] T. Wimböck, B. Jahn, and G. Hirzinger. Synergy level impedance control for multifingered hands. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 973–979, 2011.
- [99] T. Laliberte, L. Birglen, and C. Gosselin. Underactuation in robotic grasping hands. *Machine Intelligence & Robotic Control*, 4(3):1–11, 2002.
- [100] K. Hang, W. G. Bircher, A. S. Morgan, and A. M. Dollar. Hand-object configuration estimation using particle filters for dexterous in-hand manipulation. *The International Journal of Robotics Research*, 39(14):1760–1774, 2020.
- [101] H. van Hoof, T. Hermans, G. Neumann, and J. Peters. Learning robot in-hand manipulation with tactile features. In *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*, pages 121–127, 2015.
- [102] N. Elangovan, A. Dwivedi, L. Gerez, C. Chang, and M. Liarokapis. Employing IMU and ArUco Marker Based Tracking to Decode the Contact Forces Exerted by Adaptive Hands. In *2019 IEEE-RAS 19th International Conference on Humanoid Robots (Humanoids)*, pages 525–530, 2019.

- [103] H. Tjaden, U. Schwanecke, E. Schömer, and D. Cremers. A region-based gauss-newton approach to real-time monocular multiple object tracking. *IEEE transactions on pattern analysis and machine intelligence*, 41(8):1797–1812, 2018.
- [104] Y. Li, G. Wang, X. Ji, Y. Xiang, and D. Fox. Deepim: Deep iterative matching for 6d pose estimation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 683–698, 2018.
- [105] F. Chaumette, S. Hutchinson, and P. Corke. *Visual Servoing*, pages 841–866. Springer International Publishing, Cham, 2016.
- [106] A. S. Morgan, K. Hang, and A. M. Dollar. Object-Agnostic Dexterous Manipulation of Partially Constrained Trajectories. *IEEE Robotics and Automation Letters*, 5(4):5494–5501, 2020.
- [107] N. Chavan-Daffe, A. Rodriguez, R. Paolini, B. Tang, S. Srinivasa, M. Erdmann, M. T. Mason, I. Lundberg, H. Staab, and T. Fuhlbrigge. Extrinsic Dexterity: In-Hand Manipulation with External Forces. In *Proceedings of (ICRA) International Conference on Robotics and Automation*, pages 1578 – 1585, May 2014.
- [108] H. Park, J. Park, D. Lee, J. Park, M. Baeg, and J. Bae. Compliance-Based Robotic Peg-in-Hole Assembly Strategy Without Force Feedback. *IEEE Transactions on Industrial Electronics*, 64(8):6299–6309, 2017.
- [109] T. Tang, H. Lin, Yu Zhao, Wenjie Chen, and M. Tomizuka. Autonomous alignment of peg and hole by force/torque measurement for robotic assembly. In *2016 IEEE International Conference on Automation Science and Engineering (CASE)*, pages 162–167, 2016.
- [110] T. Tang, H. Lin, Y. Zhao, Y. Fan, W. Chen, and M. Tomizuka. Teach industrial robots peg-hole-insertion by human demonstration. In *2016 IEEE International Conference on Advanced Intelligent Mechatronics (AIM)*, pages 488–494, 2016.
- [111] J. Xu, Z. Hou, W. Wang, B. Xu, K. Zhang, and K. Chen. Feedback Deep Deterministic Policy Gradient With Fuzzy Reward for Robotic Multiple Peg-in-Hole Assembly Tasks. *IEEE Transactions on Industrial Informatics*, 15(3):1658–1667, 2019.
- [112] Y. Fie and X. Zhao. An Assembly Process Modeling and Analysis for Robotic Multiple Peg-in-hole. *Journal of Intelligent and Robotic Systems*, 36:175–189, 2003.
- [113] K. Zhang, J. Xu, H. Chen, J. Zhao, and K. Chen. Jamming Analysis and Force Control for Flexible Dual Peg-in-Hole Assembly. *IEEE Transactions on Industrial Electronics*, 66(3):1930–1939, 2019.

- [114] G. Schoettler, A. Nair, J. A. Ojea, S. Levine, and E. Solowjow. Meta-Reinforcement Learning for Robotic Industrial Insertion Tasks. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 9728–9735, 2020.
- [115] M. P. Polverini, A. M. Zanchettin, S. Castello, and P. Rocco. Sensorless and constraint based peg-in-hole task execution with a dual-arm robot. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 415–420, 2016.
- [116] S. Levine, N. Wagener, and P. Abbeel. Learning contact-rich manipulation skills with guided policy search. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 156–163, 2015.
- [117] C. Choi, J. Del Preto, and D. Rus. Using Vision for Pre- and Post-grasping Object Localization for Soft Hands. In D. Kulić, Y. Nakamura, O. Khatib, and G. Venture, editors, *2016 International Symposium on Experimental Robotics*, pages 601–612, Cham, 2017. Springer International Publishing.
- [118] W.-C. Chang. Robotic assembly of smartphone back shells with eye-in-hand visual servoing. *Robotics and Computer-Integrated Manufacturing*, 50:102–113, 2018.
- [119] Y. She, S. Wang, S. Dong, N. Sunil, A. Rodriguez, and E. Adelson. Cable Manipulation with a Tactile-Reactive Gripper. In *Robotics: Science and Systems*, 2020.
- [120] Z. Liu, L. Song, Z. Hou, K. Chen, S. Liu, and J. Xu. Screw Insertion Method in Peg-in-Hole Assembly for Axial Friction Reduction. *IEEE Access*, 7:148313–148325, 2019.
- [121] J. Xu, Z. Hou, Z. Liu, and H. Qiao. Compare Contact Model-based Control and Contact Model-free Learning: A Survey of Robotic Peg-in-hole Assembly Strategies, 2019.
- [122] K. Van Wyk, M. Culleton, J. Falco, and K. Kelly. Comparative Peg-in-Hole Testing of a Force-Based Manipulation Controlled Robotic Hand. *IEEE Transactions on Robotics*, 34(2):542–549, 2018.
- [123] M. A. Lee, Y. Zhu, K. Srinivasan, P. Shah, S. Savarese, L. Fei-Fei, A. Garg, and J. Bohg. Making Sense of Vision and Touch: Self-Supervised Learning of Multimodal Representations for Contact-Rich Tasks. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8943–8950, 2019.
- [124] K. Zakka, A. Zeng, J. Lee, and S. Song. Form2Fit: Learning Shape Priors for Generalizable Assembly from Disassembly. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9404–9410, 2020.

- [125] G. Thomas, M. Chien, A. Tamar, J. A. Ojea, and P. Abbeel. Learning Robotic Assembly from CAD. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3524–3531, 2018.
- [126] C. C. Beltran-Hernandez, D. Petit, I. G. Ramirez-Alpizar, and K. Harada. Variable Compliance Control for Robotic Peg-in-Hole Assembly: A Deep-Reinforcement-Learning Approach. *Applied Sciences*, 10(19):6923, Oct 2020.
- [127] S. Levine, C. Finn, T. Darrell, and P. Abbeel. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, 17(1):1334–1373, 2016.
- [128] S. Levine, P. Pastor, A. Krizhevsky, J. Ibarz, and D. Quillen. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *The International Journal of Robotics Research*, 37(4-5):421–436, 2018.
- [129] U. Viereck, A. Pas, K. Saenko, and R. Platt. Learning a visuomotor controller for real world robotic grasping using simulated depth images. In *Conference on Robot Learning*, pages 291–300. PMLR, 2017.
- [130] F. Ebert, C. Finn, S. Dasari, A. Xie, A. Lee, and S. Levine. Visual foresight: Model-based deep reinforcement learning for vision-based robotic control. *arXiv preprint arXiv:1812.00568*, 2018.
- [131] S. Bechtle, N. Das, and F. Meier. Learning Extended Body Schemas from Visual Keypoints for Object Manipulation. *arXiv preprint arXiv:2011.03882*, 2020.
- [132] L. Manuelli, Y. Li, P. Florence, and R. Tedrake. Keypoints into the Future: Self-Supervised Correspondence in Model-Based Reinforcement Learning. *CoRL*, 2020.
- [133] A. Byravan, F. Leeb, F. Meier, and D. Fox. Se3-pose-nets: Structured deep dynamics models for visuomotor control. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3339–3346. IEEE, 2018.
- [134] J. Kober, J. A. Bagnell, and J. Peters. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32(11):1238–1274, 2013.
- [135] D. Kappler, F. Meier, J. Issac, J. Mainprice, C. G. Cifuentes, M. Wüthrich, V. Berenz, S. Schaal, N. Ratliff, and J. Bohg. Real-time perception meets reactive motion generation. *IEEE Robotics and Automation Letters*, 3(3):1864–1871, 2018.

- [136] C. Mitash, R. Shome, B. Wen, A. Boularias, and K. Bekris. Task-Driven Perception and Manipulation for Constrained Placement of Unknown Objects. *IEEE Robotics and Automation Letters*, 5(4):5605–5612, 2020.
- [137] W. Gao and R. Tedrake. kPAM 2.0: Feedback Control for Category-Level Robotic Manipulation. *IEEE Robotics and Automation Letter (RA-L)*, 2020.
- [138] J. Issac, M. Wüthrich, C. G. Cifuentes, J. Bohg, S. Trimpe, and S. Schaal. Depth-based object tracking using a robust gaussian filter. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 608–615. IEEE, 2016.
- [139] M. Wüthrich, P. Pastor, M. Kalakrishnan, J. Bohg, and S. Schaal. Probabilistic object tracking using a range camera. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3195–3202. IEEE, 2013.
- [140] S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. Davison, et al. KinectFusion: real-time 3D reconstruction and interaction using a moving depth camera. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, pages 559–568, 2011.
- [141] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 23–30, 2017.
- [142] B. Calli, A. Singh, J. Bruce, A. Walsman, K. Konolige, S. Srinivasa, P. Abbeel, and A. M. Dollar. Yale-CMU-Berkeley dataset for robotic manipulation research. *The International Journal of Robotics Research*, 36(3):261–268, 2017.
- [143] B. Galerne, Y. Gousseau, and J.-M. Morel. Random phase textures: Theory and synthesis. *IEEE Transactions on image processing 2010*, 20(1):257–267, 2011.
- [144] R. B. Rusu and S. Cousins. 3D is here: Point Cloud Library (PCL). In *2011 IEEE International Conference on Robotics and Automation*, pages 1–4, 2011.
- [145] I. A. Sucas, M. Moll, and L. E. Kavraki. The open motion planning library. *IEEE Robotics & Automation Magazine*, 19(4):72–82, 2012.
- [146] R. Ma and A. Dollar. Yale OpenHand Project: Optimizing Open-Source Hand Designs for Ease of Fabrication and Adoption. *IEEE Robotics Automation Magazine*, 24(1):32–40, 2017.
- [147] J. Loncaric. Normal forms of stiffness and compliance matrices. *IEEE Journal on Robotics and Automation*, 3(6):567–572, 1987.

- [148] J. Bimbo, A. S. Morgan, and A. M. Dollar. Force-based simultaneous mapping and object reconstruction for robotic manipulation. *IEEE Robotics and Automation Letters*, 7(2):4749–4756, 2022.
- [149] M. Skubic and R. Volz. Identifying contact formations from sensory patterns and its applicability to robot programming by demonstration. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems. IROS '96*, volume 2, pages 458–464 vol.2, 1996.
- [150] Y. Xu and R. Paul. A robot compliant wrist system for automated assembly. In *Proceedings., IEEE International Conference on Robotics and Automation*, pages 1750–1755 vol.3, 1990.
- [151] *Remote Center of Compliance Reconsidered*, volume Volume 2A: 24th Biennial Mechanisms Conference of *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, 08 1996.
- [152] A. S. Morgan, B. Wen, J. Liang, A. Boularias, A. M. Dollar, and K. Bekris. Vision-driven Compliant Manipulation for Reliable; High-Precision Assembly Tasks. In *Proceedings of Robotics: Science and Systems*, Virtual, July 2021.
- [153] U. Thomas and F. M. Wahl. *Assembly Planning and Task Planning — Two Prerequisites for Automated Robot Programming*, pages 333–354. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.
- [154] A. Farahat, B. Graves, and J. Trinkle. Identifying contact formations in the presence of uncertainty. In *Proceedings 1995 IEEE/RSJ International Conference on Intelligent Robots and Systems. Human Robot Interaction and Cooperative Robots*, volume 3, pages 59–64 vol.3, 1995.
- [155] X. Cheng, E. Huang, Y. Hou, and M. T. Mason. Contact mode guided sampling-based planning for quasistatic dexterous manipulation in 2d. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6520–6526, 2021.
- [156] S. Cabras, M. E. Castellanos, and E. Staffetti. Contact-state classification in human-demonstrated robot compliant motion tasks using the boosting algorithm. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 40(5):1372–1386, 2010.
- [157] K. Hang, A. S. Morgan, and A. M. Dollar. Pre-grasp sliding manipulation of thin objects using soft, compliant, or underactuated hands. *IEEE Robotics and Automation Letters*, 4(2):662–669, 2019.

- [158] K. Kimble, K. Van Wyk, J. Falco, E. Messina, Y. Sun, M. Shibata, W. Uemura, and Y. Yokokohji. Benchmarking protocols for evaluating small parts robotic assembly systems. *IEEE Robotics and Automation Letters*, 5(2):883–889, 2020.
- [159] S. Sajjan, M. Moore, M. Pan, G. Nagaraja, J. Lee, A. Zeng, and S. Song. Clear Grasp: 3D Shape Estimation of Transparent Objects for Manipulation. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3634–3642. IEEE, 2020.