Design and Control of Roller Grasper V3 for In-Hand Manipulation

Shenli Yuan, Lin Shao, Yunhai Feng, Jiatong Sun, Teng Xue, *Graduate Student Member, IEEE*, Connor L. Yako, Jeannette Bohg, *Member, IEEE*, and J. Kenneth Salisbury, *Life Member, IEEE*

***Abstract*—Robot in-hand manipulation is an important skill for robots to carry out sophisticated tasks that require moving the grasped object within hand. In this work, we present the Roller Grasper V3, a nonanthropomorphic robot grasper with a steerable roller on each of its four fingertips, and a manipulation architecture that enables the Roller Grasper V3 to achieve full 6-DoF manipulation of the grasped object in $SE(3)$. The manipulation architecture consists of a high-level planner that searches for a feasible path with waypoints for the object to be manipulated, and a low-level control policy that is used to navigate the object in between the adjacent waypoints. The method was experimentally validated on the Roller Grasper V3 to manipulate multiple objects with different geometries and topologies.**

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

I. Introduction

THE robotics research community has long desired to develop robots capable enough to seamlessly operate in our unstructured world. While many advancements are needed to ultimately achieve this goal, one area of particular importance is robot dexterity. Robots acting in the real world will need to perform rich manipulation tasks on a variety of objects. Many of these high-level tasks will rely on the robot's ability to perform in-hand manipulation, specifically reorientation, and translation primitives, all while maintaining a stable grasp. Examples include object inspection, assembly, packing, etc. Out of all the grasping and manipulation tasks facing a robot, in-hand manipulation is among those that require the most dexterity.

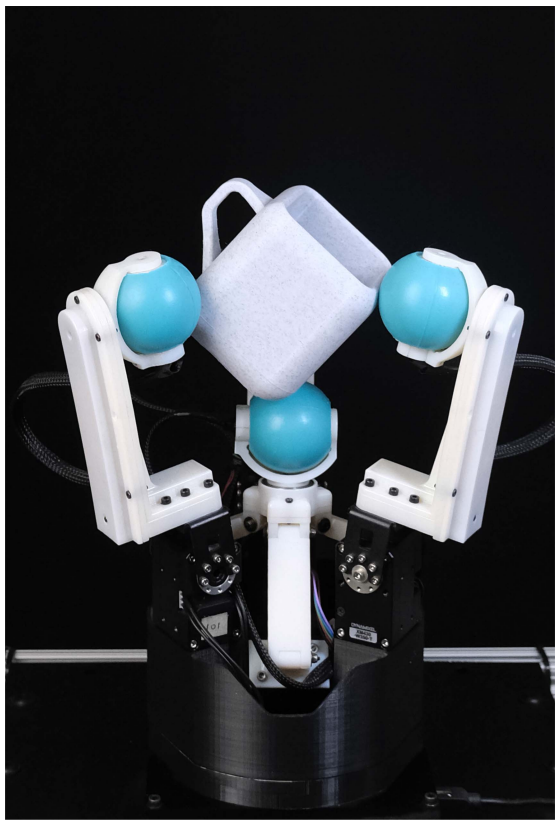


Fig. 1. Prototype of the Roller Grasper V3 holding a 3D-printed mug.

We have designed a new generation of Roller Grasper (see Fig. 1) with articulated and actively driven spherical rollers on each of its four fingertips.

The most significant advantage of incorporating rolling contacts for in-hand manipulation is the ability of the fingers to change contact locations without the need for finger gaiting [1], which can be difficult to control for a real robot. The new Roller Grasper V3 improves upon the previous Roller Grasper V2 three-finger design [2] through the addition of a fourth finger that acts as a movable, steerable palm. This active palm significantly improves grasp stability across many grasper poses, leading to a larger set of stable grasp poses while moving the object within hand.

We investigated how to enable the grasper to move a grasped object from an arbitrary initial pose to an arbitrary goal pose. This is a challenging task due to the diversity of object

Received 6 June 2024; revised 4 August 2024; accepted 10 August 2024. Date of publication 5 September 2024; date of current version 19 September 2024. This article was recommended for publication by Associate Editor Ahmed H. Qureshi and Editor Sven Behnke upon evaluation of the reviewers' comments. *(Corresponding author: Shenli Yuan.)*

Shenli Yuan, Connor L. Yako, Jeannette Bohg, and J. Kenneth Salisbury are with the Stanford University, Stanford, CA 94305 USA (e-mail: shenliy@stanford.edu; clyako@stanford.edu; bohg@stanford.edu; kenneth.salisbury@gmail.com).

Lin Shao is with the National University of Singapore, Singapore 119077 (e-mail: linshao@nus.edu.sg).

Yunhai Feng is with the Cornell University, Ithaca, NY 14850 USA (e-mail: yf428@cornell.edu).

Jiatong Sun is with the University of Pennsylvania, Philadelphia, PA 19104 USA (e-mail: sjtupenn1998@gmail.com).

Teng Xue is with the Idiap Research Institute, EPFL, 1015 Lausanne, Switzerland (e-mail: teng.xue@epfl.ch).

This article has supplementary downloadable material available at https://doi.org/10.1109/TRO.2024.3454388, provided by the authors.

Digital Object Identifier 10.1109/TRO.2024.3454388

geometries, the infinite number of combinations of initial and target poses, and the infinite number of possible paths between these poses in SE(3) space. We adopted a hybrid framework that leverages both traditional model-based and model-free approaches. Unlike existing hybrid learning-and-control approaches, which typically focus on learning local dynamics models [3], or residual policy learning [4], [5] for short-horizon tasks [6], we formulated the in-hand manipulation task as a path planning problem at a high level, integrating it with a low-level controller within a hierarchical manipulation framework. The high-level planner identifies a feasible path with waypoints between the initial and target object poses, while the low-level controller guides the object between these waypoints. Different from the approach in [7] that uses the object poses as input, our planner processes raw point clouds of objects. Our pipeline can generalize to both novel goals and novel objects, demonstrating its effectiveness on a real robotic hand manipulating various 3-D objects.

The contributions of this work are twofold: 1) an increased object range of motion from Roller Grasper V2 by incorporating an active palm that augments grasp stability, and 2) a hierarchical manipulation framework that allows for object manipulation in SE(3). We applied this framework on the Roller Grasper V3 to transform various objects including a cube, a cube with an opening, a rectangular prism, a mug, and a rectangular prism with a handle.

II. Related Work

A. Robot Hand Design

An incredible amount of hands have been designed over the course of robotics research. An extensive review of robotic hand applications as well as emerging trends in actuation and design can be found in [8]. This review features both complex and simpler grippers, but it also shows that there has been a trend toward simplified design. While high degree of freedom (DoF) anthropomorphic grippers, such as those found in [9], [10], [11], are highly dexterous, their complexity leads to their lack of use outside of research.

Nonanthropomorphic approaches have realized grippers with fewer, carefully selected DoFs, many of which are underactuated. Despite the lack of controllability associated with underactuated mechanisms, these grippers can still perform dexterous tasks. For example, the GR2 Gripper could achieve object reorientations greater than 90° [12]; the iHY Hand could perform a set of manipulation tasks such as reorienting thin objects on flat surfaces and repositioning an object into a power grasp [13], as well as stable precision manipulation of an object [14]; the four-fingered gripper presented in [15] could perform finger-gaiting and object reorientation. Similar examples of dexterous underactuated grippers can be seen in [16], [17], [18], [19], [20] to name a few.

Gripper dexterity can also be achieved using active surfaces in place of longer serial-chain DoFs. These active surfaces allow the gripper to transform the grasped object through locally imparted motions at the contact points. Inclusion of active surfaces as a means of increasing manipulation capability has been explored in [21], [22], [23], [24]. In these works, the orientation of the active surfaces are fixed and thus the object motion is restricted to the plane of the gripper, rather than being able to move spatially. Previous works [2], [25] have lifted these restrictions by controlling the orientation of the active surfaces.

B. In-Hand Manipulation

Developing robust policies for in-hand manipulation has been a long-standing challenge in robotics. Analytical strategies in general have assumed either fixed contacts between the object and the fingers [26], sliding contacts [27], [28], [29], and/or rolling contacts [30]. They also typically assume accurate contact models, which is a strong assumption for a real-world manipulation system. Rather than modeling the different types of contact or opting for computationally expensive planning of when to switch contacts in multifingered in-hand manipulation, many have opted to use deep reinforcement learning (DRL) [31], [32]. However, DRL approaches require a tremendous amount of training episodes and a carefully designed reward function to be successful. An alternative to selecting one of the two aforementioned approaches is to instead combine them in a hierarchical manner. Hybrid approaches leverage a model-based, low-level manipulation primitive controller in conjunction with a high-level model-free planner. A closed-loop controller at the low-level combined with a generalizable planner at the high level has demonstrated success [7]. We will briefly review different algorithmic approaches to in-hand manipulation in the following paragraphs.

1) Model-Based Control and Planning: Model-based in-hand manipulation approaches typically assume known kinematics, dynamics, and contact models. One major challenge for model-based approaches is the high dimensionality of the state and action space for tasks that require a sequence of actions to reach the target object pose. To address the challenge, action primitives such as grasping, finger sliding [33] or finger gaiting [34] are developed and integrated into hierarchical control schemes [1], [35], [36]. They break down the in-hand manipulation tasks into multiple sublevel tasks and address control of the object at each of these sublevels. Other works [37], [38] formulate the manipulation problem as a graph searching problem and generate the manipulation sequence by selecting a feasible or optimal path. In-hand manipulation has also been tackled with contact-invariant optimization [39] and model-predictive control [14], [40]. Another method is an energy-based approach [41], [42], [43]. The potential energy of the system is modified by changing the configuration of the hand in order to drive an object to a specified pose in the hand frame. However, these strong assumptions, such as fully known, deterministic transition models in those model-based approaches, limit their potential applications in the real-world setting.

2) Model-Free Reinforcement Learning: Unlike model-based approaches, model-free approaches assume limited or no prior knowledge about the robotic hand or object and learn to manipulate the object through reinforcement learning. Leveraging complex neural network models, DRL algorithms have been applied to solve a Rubik's Cube [44], manipulate a pen [45], and

train a policy to roll and reorient a cylinder using tactile sensory data as input [46]. These model-free approaches learn a mapping between observations of the object and control commands of the robot hand [32], [47], [48]. Tao et al. [49] proposed the teacher–student training paradigm. They first trained teacher policies using information that is easy to gather in simulation, such as velocities, as inputs and used the teacher policies to train student policies with object point clouds as inputs. These model-free RL approaches require a large number of training samples. To apply data-hungry DRL algorithms, researchers have implemented algorithms only in robotic simulations or proposed various approaches, such as domain randomization [44], [50], to reduce the sim-to-real gap for robotic manipulation tasks in real-robot experiments.

3) Hybrid Learning-and-Control: A hybrid framework combines the advantages of both traditional model-based and model-free approaches. Residual policy learning (RPL) [4], [5] provides a way to enhance a given base control policy by learning additive, corrective actions using RL. RPL has been applied to in-hand manipulation tasks [6], improving both sample efficiency and exploration and leading to policies that can outperform classical approaches as well as pure RL. Another line of work combines the model-based hierarchical control framework with learned models [3]. Unlike these hybrid learning-and-control approaches, our system is designed for dexterous manipulation in long-horizon tasks. A hybrid approach was used in [7], where the DRL policy takes as input the current and target poses and selected manipulation primitives to be executed by the low-level controller to manipulate known 2-D objects. Our approach leverages a model-based low-level controller that executes a manipulation primitive that reposes an object to a desired goal pose provided by a high-level model-free planner. Our planner also differs from [7] by taking raw point clouds of the object as input and predicting whether there is at least one feasible path on which the low-level controller could successfully move the object. This approach can generalize to novel objects. In addition, we have demonstrated the effectiveness of our approach on a real robotic hand to manipulate a variety of 3-D objects, including novel objects.

III. Grasper Design

A. Design

The grasper consists of four fingers, each having three DoFs (see Fig. 2). Three of the four fingers are identical to one another, placed 120° apart at the base. These three fingers are also kinematically identical to the ones on the Roller Grasper V2. The first DoF of these fingers is a revolute joint directly driven by a Robotis Dynamixel XM430-W350 actuator. The second joint, orthogonal to the first joint, is driven by a micro dc motor (Servocity No.638099) through a timing belt; this joint can be pitched up to 360°. Note that this limit is only enforced in software to prevent the cables of the motors inside the rollers from winding up. The final DoF is actuated by the same motor as the second joint and is located inside the roller assembly for a compact form-factor and unbounded rotation. The manipulation algorithm sends position commands to the motors and receives real-time motor positions through a Teensy 3.6 microcontroller over a custom communication protocol. The microcontroller is also used for the PD control of the micro dc motors. The Dynamixel motors each run local PD control and their communication with the manipulation algorithm is handled using Dynamixel SDK.

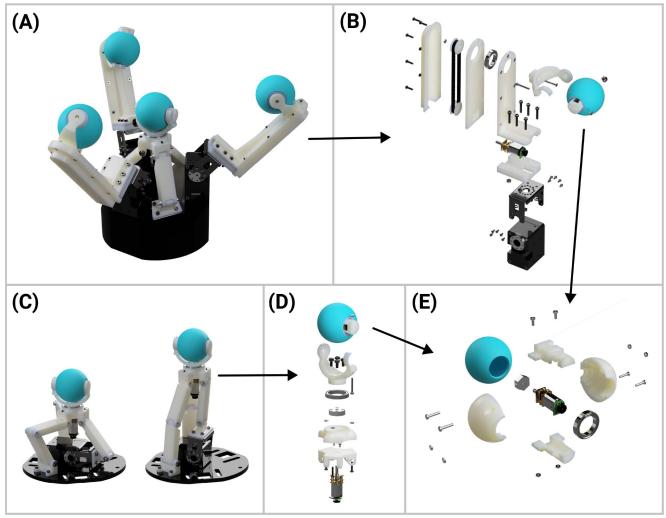


Fig. 2. CAD renderings of the mechanical design of Roller Grasper V3. (A) Fully assembled hand. (B) Exploded view showing the pivot and base joints assembly for the three fingers with revolute base joint. (C) Parallel mechanism used to achieve the prismatic joint at the center of the hand (Lowest position shown at left and highest position shown at right). (D) Exploded view of the pivot joint attached on top of the parallel mechanism. (E) Exploded view of the roller assembly that is used on all four fingers.

The fourth finger (also referred to as an active palm) is located at the center of the base and moves linearly along the base's normal axis. The prismatic joint is driven by a parallel one-DoF linkage consisting of six revolute joints, one of which is actively driven by a Dynamixel actuator while the other five are passive. Similar to the other three fingers, this finger also has DoFs for steering and rolling. The axis of the steering motor is colinear with the active palm's prismatic joint axis, and is directly driven by the same type of micro dc motors mentioned above. The roller assembly is identical to the ones on the other fingertips.

There are several reasons for opting for a parallel mechanism over a linear actuator to implement the prismatic joint. The first reason is to avoid the binding issues that often occur with linear actuators. During in-hand manipulation, the prismatic joint is frequently subjected to sideloading and eccentric loading, conditions that are highly undesirable for linear actuators. Although adding additional linear motion stages could alleviate some loading issues, this would introduce more friction into the system, rendering this DoF nonbackdrivable. This brings us to the second reason for choosing a parallel mechanism: it allows the prismatic joints to be backdrivable outside of singularity positions. This feature enables secure grasping of the object during manipulation with simple position control. The final reason is that parallel joints allow the links to fold when the roller is in its lowest position, significantly reducing the overall height of the system. In contrast, a linear actuator typically has a body length

much greater than its stroke length when fully retracted, resulting in a much taller form factor for the hand. While the parallel mechanism effectively minimizes the vertical dimension of the gripper, it can result in a relatively large horizontal footprint, particularly when the prismatic joint is fully retracted. To address this, link length optimization using MATLAB's `fmincon` was employed, primarily to minimize the horizontal spread of the parallel mechanism. In addition, this optimization enhances the resolution of the prismatic joint movements, as shorter links undergo a larger rotation to accomplish the same vertical travel. The final optimized parallel mechanism has a footprint that is comparable to the rest of the gripper.

It is important to highlight that the objective of this research is to offer an alternative to conventional anthropomorphic hands, which may prove advantageous in particular scenarios. Our intention is not to propose that this approach should supplant other types of robotic hands entirely. Rather, we envision this work as a foundational contribution to the field of autonomous manipulation using rolling contact. We anticipate that the specific form-factor of the hands could be tailored to suit distinct applications, thereby broadening the scope and utility of our findings. This adaptability could potentially lead to innovations in how robotic hands are designed and utilized across various industries, enhancing both efficiency and functionality in automated systems.

B. Grasp Quality Comparison Between Roller Grasper V2 and V3

The main reason for adding a fourth finger to the existing Roller Grasper V2 was to increase the quality of grasps during in-hand manipulation. Since the majority of in-hand manipulation techniques, including those that utilize rolling contact and finger-gaiting, rely on precision grasps, grasp stability is inevitably compromised compared to that of a power grasp. Adding a fourth finger can help mitigate this issue by enhancing the overall stability of certain grasp poses and enabling additional stable grasp poses during manipulation. Since we have taken a quasi-static approach to solve the manipulation problem, we are assuming the object is in static equilibrium at every step in the manipulation sequence. Below, we provide a simple mathematical proof that the added fourth finger is beneficial in terms of grasp quality by incorporating the L_1 grasp quality criterion [51]. We adopt the notation from [52].

A grasp configuration g can be described by m contact points on the object surface $c_1, c_2, \ldots, c_m \in \mathcal{R}^3$ with corresponding inward pointing unit normal vectors $n_1, n_2, \ldots, n_m \in \mathcal{S}^2 = \{x \in \mathcal{R}^3 \mid \|x\| = 1\}$, and the object's center of mass denoted as $z \in \mathcal{R}^3$.

Given a grasp configuration g, the space of wrenches that can be applied to the object is bounded by the convex hull denoted as $W(g)$, spanned by $(w_1, w_2, \ldots, w_m)$ and with contact force $\sum_{i=1}^{m} \|f_i^n\| \leq 1$. Here, $w_i = (f_i, (c_i - z) \times f_i) \in \mathcal{R}^6$ is the wrench exerted at the object surface c_i about the object's center of mass with contact force f_i. We denote the grasp quality of a given g as Volume$(W(g))$, which is the volume of the wrench space $W(g)$ [53].

TABLE I
LIST OF NOTATIONS

Symbol	Explanation
$\mathbf{p}$	object pose
$\mathbf{p}_{init}$	object initial pose
$\mathbf{p}_{goal}$	object goal pose
$\mathcal{O}_i$	the object with an index of i
$q_{obj,d}$	object desired orientation in quaternion representation
$q_{obj,c}$	object actual orientation in quaternion representation
X	object position
$\mathcal{R}$	object orientation
Λ	the tree constructed by the high-level planner
k_X	weight of position distance
D_X	position distance
$k_{\mathcal{R}}$	weight of orientation distance
$D_{\mathcal{R}}$	orientation distance
$\delta X_{contact}$	contact motion*
δX_{cb}	contact motion resulted from base joint movement*
δX_{cr}	contact motion resulted from roller joint movement*
Z_2	pivot joint rotational axis*
$\hat{n}_{con}$	normal axis which defines the contact plane*

* See [2] for further details.

Given the same object, the grasp configuration for Roller V2 is denoted as g_2 with contact points c_1, c_2, c_3, and the grasp configuration for Roller V3 is denoted as g_3 with contact points c_1, c_2, c_3, c_4. Note that c_1, c_2, c_3 are the same contact points for both graspers. If the additional contact wrench introduced by c_4 lies inside $W(g_2)$, Volume$(W(g_3))$ is unchanged. Otherwise, Volume$(W(g_3))$ will become larger. This indicates that while the addition of the fourth finger does not universally enhance grasp quality across all poses, it does improve it in various scenarios and can stabilize previously unstable grasps. It is important to point out that this analysis is independent of gravitational effects. The addition of a fourth finger is not intended to simulate a human palm for merely supporting objects on top, but rather to enhance the quality of the grasp and increase the number of stable grasp poses available throughout the manipulation process. This improvement allows for more effective searching of feasible paths during manipulation, thus broadening the Roller Grasper V3's capabilities in handling a wide variety of objects.

IV. MANIPULATION ALGORITHM

A. Overview

In this work, we investigate how to provide the grasper with the ability to transform a random object $\mathcal{O}_i$ from an arbitrary initial pose $\mathbf{p}_{\text{init}} \in \text{SE}(3)$ to an arbitrary goal pose $\mathbf{p}_{\text{goal}} \in \text{SE}(3)$. (The notations used in this article are summarized in Table I.) There are an infinite number of tasks our gripper may be asked to perform due to the diversity of possible object geometries and the infinitely many combinations of arbitrary initial and target poses in SE(3) space. Our method leverages both a low-level controller and a high-level planner to accomplish this task. The low-level controller approximates the object as a sphere and attempts to directly manipulate the object between its current pose $\mathbf{p}_a$ and a target pose $\mathbf{p}_b$. As a result, it typically only accomplishes the task in the absence of complicated topology or severe geometry changes along the slerp interpolation path region between $\mathbf{p}_a$

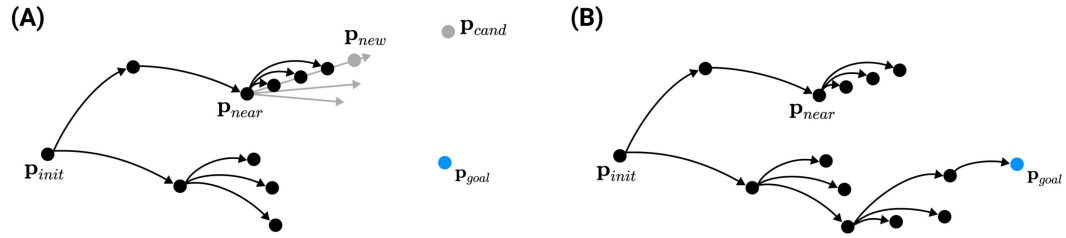


Fig. 3. (A) High-level planner illustration: a 2-D representation of the planning that takes place in SE(3) space. Black markers and arrow represent nodes and edges on the tree. In each of the steps, the planner would sample a candidate node in space $\mathbf{p}_{\text{cand}}$ and find the nearest node on the existing tree $\mathbf{p}_{\text{near}}$. The grey arrows represent three different exploration directions from $\mathbf{p}_{\text{near}}$ (explained in high-level planning section in detail). The planner then expands the tree greedily in each direction until the next node is not reachable, and all the reachable nodes will be added as the child nodes of $\mathbf{p}_{\text{near}}$. (B) After sufficient exploration, the planner finds a feasible path to manipulate the grasped object from its initial pose $\mathbf{p}_{init}$ to the target pose $\mathbf{p}_{\text{goal}}$ by going through a series of waypoints.

and $\mathbf{p}_b$. The high-level planner can refine this manipulation path by finding stable intermediate poses, rather than blindly steering an object between two poses like the low-level controller. Using both the high-level path planner and the low-level controller, we have developed an in-hand manipulation architecture to find a feasible path to accomplish a given manipulation task. In this section, we first introduce our high-level planning algorithm and then discuss our low-level controller.

B. High-Level Planning

This section formulates the algorithm proposed for high-level planning based on the rapidly exploring random tree (RRT) algorithm [54], which is also illustrated in Fig 3. Given an initial pose $\mathbf{p}_{\text{init}} = (X_{\text{init}}, \mathcal{R}_{\text{init}})$ and a desired goal pose $\mathbf{p}_{\text{goal}} = (X_{\text{goal}}, \mathcal{R}_{\text{goal}})$, where X and $\mathcal{R}$ denote the position and orientation of the object, respectively, Algorithm 1 outputs a feasible path $P = \langle \mathbf{p}_{\text{init}}, \mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_{\text{goal}} \rangle$ from $\mathbf{p}_{\text{init}}$ to $\mathbf{p}_{\text{goal}}$ if it can be found within K iterations.

The high-level planner first has the search tree Λ initialized with a single root node $\mathbf{p}_{\text{init}}$. It then iteratively adds new reachable nodes to Λ until one of the following termination conditions are met: 1) a feasible path is found, or 2) the maximum number of iterations is reached. In each iteration, a candidate pose $\mathbf{p}_{\text{cand}}$ is first generated by CANDIDATEPOSE() using the following heuristic sampling strategy: with a probability P_{goal}, $\mathbf{p}_{\text{cand}}$ will be identical to $\mathbf{p}_{\text{goal}}$. This is to encourage the planner to explore a path directly toward the goal in case a feasible path already exists and no further iterations will be necessary. On the other hand, a new candidate pose $\mathbf{p}_{\text{cand}} = (X_{\text{cand}}, \mathcal{R}_{\text{cand}})$ will be sampled with a probability of $\overline{P}_{\text{goal}} = 1 - P_{\text{goal}}$. In this case, X_{cand} is uniformly sampled within a 4cm $\times$ 4cm $\times$ 3cm volume in R^3, and $\mathcal{R}_{\text{cand}}$ is uniformly sampled in SO(3). We then apply a biased sampling technique on $\mathbf{p}_{\text{cand}}$ toward the goal. Specifically, if the newly sampled $\mathbf{p}_{\text{cand}}$ is further away from $\mathbf{p}_{\text{goal}}$ compared to its parent node $\mathbf{p}_{\text{near}}$, there is a probability of P_{rej} that this particular $\mathbf{p}_{\text{cand}}$ will be directly rejected and the planner will resample a new $\mathbf{p}_{\text{cand}}$. This design allows the planner to converge faster and select a shorter path with a larger probability, which results in a more efficient manipulation. After $\mathbf{p}_{\text{cand}}$ is sampled, we select a $\mathbf{p}_{\text{near}}$ using the function NEARESTNODE$(\mathbf{p}_{\text{cand}}, \Lambda)$. $\mathbf{p}_{\text{near}}$ is an existing node on Λ with the shortest distance to $\mathbf{p}_{\text{cand}}$, and the distance metric is formulated as

$$\rho(\mathbf{p}_1, \mathbf{p}_2) = k_X D_X(\mathbf{p}_1, \mathbf{p}_2) + k_{\mathcal{R}} D_{\mathcal{R}}(\mathbf{p}_1, \mathbf{p}_2) \tag{1}$$

where D_X and $D_{\mathcal{R}}$ are position distance and orientation distance, respectively, and k_X and $k_{\mathcal{R}}$ are their weights, respectively. The magnitudes of k_X and k_R were selected so that the distance terms were comparable in our applications. With the $\mathbf{p}_{\text{goal}}$, $\mathbf{p}_{\text{cand}}$, and $\mathbf{p}_{\text{near}}$ all determined, we could compute their positions relative to one another in SE(3).

After $\mathbf{p}_{\text{cand}}$ is sampled, the planner starts to explore new nodes to be added to the existing Λ by navigating from $\mathbf{p}_{\text{near}}$

Algorithm 1: High-Level Planner.

Input: initial pose $\mathbf{p}_{init}$, goal pose $\mathbf{p}_{goal}$, maximum number of iterations K
Output: a feasible path from $\mathbf{p}_{init}$ to $\mathbf{p}_{goal}$ if it is found within K iterations, and NONE otherwise
INIT$(\Lambda, \mathbf{p}_{init})$
for $k \leftarrow 1$ to K **do**
  $\mathbf{p}_{cand} \leftarrow$ CANDIDATEPOSE()
  $\mathbf{p}_{near} \leftarrow$ NEARESTNODE$(\mathbf{p}_{cand}, \Lambda)$
  if REJECTCANDIDATE$(\mathbf{p}_{cand}, \mathbf{p}_{near}, \mathbf{p}_{goal})$ **then**
    continue
  end if
  for direction $d \leftarrow$ `weighted`, `rotation`, `translation` **do**
    for $\ell \leftarrow 1$ to max_steps **do**
      $\mathbf{p}_{new} \leftarrow$ STEP$(\mathbf{p}_{near}, d, \ell)$
      if $\mathcal{F}(\mathbf{p}_{near}, \mathbf{p}_{new}) > thresh$ **then**
        ADDNODE$(\Lambda, \mathbf{p}_{near}, \mathbf{p}_{new})$
        if $\mathcal{F}(\mathbf{p}_{new}, \mathbf{p}_{goal}) > thresh$ **then**
          return GENERATEPATH$(\Lambda, \mathbf{p}_{new}, \mathbf{p}_{goal})$
        end if
      else
        break
      end if
    end for
  end for
end for
return NONE

to $\mathbf{p}_{\text{cand}}$ in small increment steps. We take three different directions of exploration by considering X_{cand} and $\mathcal{R}_{\text{cand}}$ separately: the first direction is to perform translation only where the exploration will be a series of incremental translations from $\mathbf{p}_{\text{near}}$; the second direction is to perform rotation only where the exploration will be a series of incremental rotations from $\mathbf{p}_{\text{near}}$; and the third direction is to have incremental transformations (weighted combinations translation and rotation) from $\mathbf{p}_{\text{near}}$. These three expansion directions can also be viewed as navigating from $\mathbf{p}_{\text{near}} = (X_{\text{near}}, \mathcal{R}_{\text{near}})$ towards $(X_{\text{cand}}, \mathcal{R}_{\text{near}})$, $(X_{\text{near}}, \mathcal{R}_{\text{cand}})$, and $(X_{\text{cand}}, \mathcal{R}_{\text{cand}})$, respectively. The multidirectional exploration results in a more expansive tree and a better search coverage, and subsequently, better manipulation results. Our method shares some similarities with [55] in that we both have multidirectional exploration during expansion.

For each exploration direction, the planner takes corresponding steps toward that direction. Each new step taken would result in a new node $\mathbf{p}_{\text{new}}$ in space, and the planner then verifies whether $\mathbf{p}_{\text{new}}$ is reachable from $\mathbf{p}_{\text{near}}$ using the function $\mathcal{F}(\mathbf{p}_{\text{near}}, \mathbf{p}_{\text{new}})$, which is a pretrained neural network classifier. Note that we choose $\mathbf{p}_{\text{near}}$ as the parent node of all the newly added nodes (a star configuration) instead of having all the new nodes connected one after another starting from $\mathbf{p}_{\text{near}}$ (a daisy-chain configuration). The star configuration reduces the number of nodes in the resulting path, allowing the low-level controller to go through fewer steps. $\mathbf{p}_{\text{new}}$ will be added to the tree Λ with an edge $(\mathbf{p}_{\text{near}}, \mathbf{p}_{\text{new}})$ by $\textsc{AddNode}(\Lambda, \mathbf{p}_{\text{near}}, \mathbf{p}_{\text{new}})$ if $\mathbf{p}_{\text{new}}$ is reachable from $\mathbf{p}_{\text{near}}$, i.e., $\mathcal{F}(\mathbf{p}_{\text{near}}, \mathbf{p}_{\text{new}})$ is larger than certain threshold. The planner would continue to expand toward the given direction with the same step size as long as each of the resulting $\mathbf{p}_{\text{new}}$ is reachable from $\mathbf{p}_{\text{near}}$. Each time when a $\mathbf{p}_{\text{new}}$ is added to Λ, the planner checks if $\mathbf{p}_{\text{goal}}$ is directly reachable from $\mathbf{p}_{\text{new}}$, and if so, a feasible path will be generated marking the end of the search. Otherwise, the exploration toward a specific direction terminates when a $\mathbf{p}_{\text{new}}$ cannot be reached from $\mathbf{p}_{\text{near}}$ or when the maximum number of exploration steps is reached.

1) Reachability Between Poses: We trained a neural network model to predict whether an arbitrary pose $\mathbf{p}_b$ is reachable from another arbitrary pose $\mathbf{p}_a$ $[\mathcal{F}(\mathbf{p}_a, \mathbf{p}_b)]$. As mentioned earlier, the neural network takes the point clouds of the two poses as input and outputs the probability that the object can be moved from $\mathbf{p}_a$ to $\mathbf{p}_b$ via the low-level controller. In order to train this neural network model, we collected a large amount of data by implementing the low-level control policy in the MuJoCo physics simulation environment. For each data sample, the object was transported using the low-level controller from a starting pose $\mathbf{p}_a$ to a target pose $\mathbf{p}_b$. This produced the data example $((\mathbf{p}_a, \mathbf{p}_b), y)$, where $y = 1$ if the goal pose was reached within a certain empirically determined amount of time and without significantly slipping during the transformation, and 0 otherwise. Slipping was detected in the simulation when the object had a large velocity spike in a short amount of time.

The poses in the data were sampled as follows: the orientations were randomly sampled in $\text{SO}(3)$ while the positions were uniformly sampled in a $4\text{cm} \times 4\text{cm} \times 3\text{cm}$ volume. For each of the sampled pairs $(\mathbf{p}_{\text{init}}, \mathbf{p}_{\text{goal}})$, we needed to perform a simulated run of the low-level controller transporting the object from $\mathbf{p}_{\text{init}}$ to $\mathbf{p}_{\text{goal}}$, thereby determining the outcome which can be computationally expensive. To make the sampling process more efficient, we performed data augmentation by exploiting the rotational symmetries of the objects. For example, because the cube has 24 rotational symmetries in total, the number of data points collected by transforming the cube can be increased by 24 times. Similarly, the rectangular prism and the open cube each has four rotational symmetries. Therefore, the number of data points collected using these objects can be increased accordingly.

Another challenge was that the low-level controller can successfully transport objects only between a small percentage of sampled pose pairs. Therefore, the dataset was imbalanced. We mitigated this by collecting additional samples near previously collected positive data samples. For each previously collected positive data sample $((\mathbf{p}_{\text{init}}, \mathbf{p}_{\text{goal}}), 1)$, we sampled new pairs of starting and target poses by adding noise to $\mathbf{p}_{\text{init}}$ and $\mathbf{p}_{\text{goal}}$. The low-level controller was run using these newly sampled poses resulting in a more balanced data set, because data sampled near the existing positive data points are more likely to be positive as well.

There were five objects used in data collection: a cube, two rectangular prisms, an open cube and a mug. Out of all the data collected, 90% were used for training and 10% for validation. The tests were carried out through manipulation tasks on the real hardware.

We adopted PointNet [56] to extract features from raw point clouds. The features were then fed into a *Multilayer Perceptron* (MLP) to predict classification scores regarding reachability between poses.

C. Low-Level Control

The low-level controller is similar to the "handcrafted policy" introduced in Roller Grasper V2, with a few modifications that improve its generalizability. The purpose of the low-level controller is to manipulate the object grasped between two poses in an "aim-and-roll" fashion. This can be viewed as navigating the rollers from their initial contact locations on the object, to the desired final contact locations by simultaneously aligning the pivot joint to the direction of the desired contact location and rolling toward that direction.

The general formulation of the low-level controller is to 1) calculate desired object motion based on the current and desired poses of the object (in our case, the desired motion is simply a fraction of the difference between the current and desired poses of the object), 2) assuming the object is a sphere, infer the position of the sphere using the roller positions, and then calculate the contact location of the object on each roller from the inferred object position and roller positions, 3) calculate the desired contact motion based on the rolling without slipping condition, and 4) calculate the desired joint motion by projecting the contact motion to the rolling and base joints, and aligning the pivot joint to the contact motion. A graphical demonstration of the low-level controller is demonstrated in Fig. 4(A)–(G).

Most of the detailed implementations of the low-level controller are similar to the ones presented in (1)–(11) in [2]. There

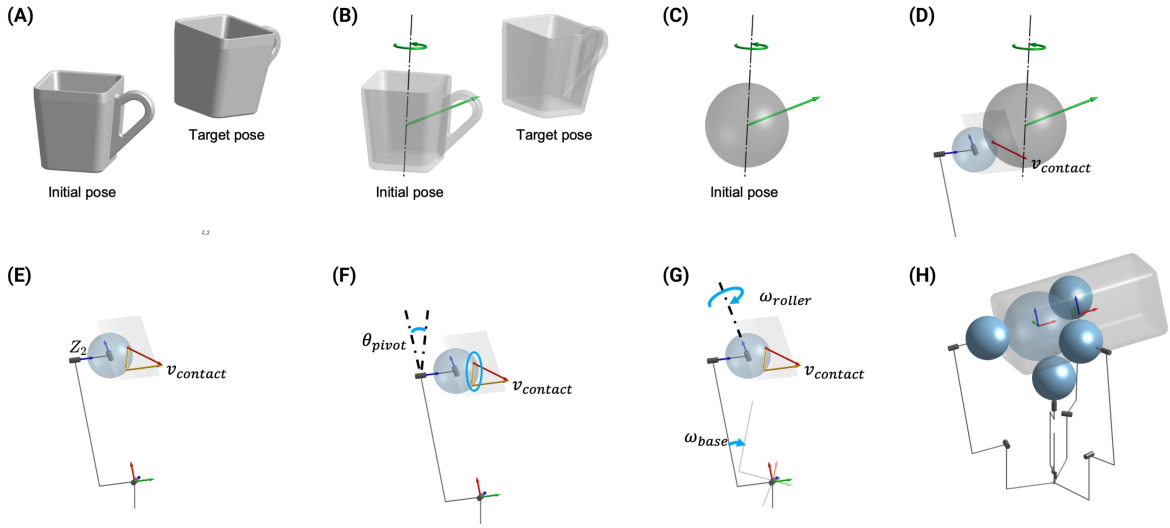


Fig. 4. (A)-(G) demonstrates how the low-level controller computes joint positions/velocities. (A) the low-level controller is manipulating the grasped object from an initial pose to a target pose; (B) the policy takes the interpolation of the target pose relative to the initial pose. (C) It assumes the object as a sphere, and (D) subsequently computes the contact location to each of the rollers and the contact velocity. (E) The contact velocity then gets projected to two directions, one being the result of base joint motion, another being the result of the rolling motion (assuming the direction of rolling is aligned by the pivot joint). (F) It calculates the angle of the pivot joint to align the rolling direction, and (G) computes the velocities of the roller and the base joint based on the two projected velocities of v_{contact}. (H) A demonstration of why the center of the assumed sphere might not be the geometrical center of the grasped object.

are a few differences of the low-level controller compared to the handcrafted policy in Roller Grasper V2: 1) There are four rollers instead of three. 2) Instead of using the geometric center of the object as the center of the assumed sphere, we inferred the sphere center using the positions of the four rollers. 3) The limit of the pivot joint increased from previous $\pm 90^\circ$ to $\pm 180^\circ$. The following derivation applies to each of the fingers including the fourth finger with a prismatic base joint. Therefore, we used subscripts 1, 2, and 3 to represent the base joint, pivot joint and roller joint, respectively, and leave out the subscripts A, B, C, and D, which represent different fingers.

Given the desired object pose and the current object pose, we set the desired motion of the grasped object to be a fraction of the difference between its current and desired poses. In this analytical process, because the object is assumed to be a sphere of fixed diameter, the location of this sphere can be inferred from the roller positions under the assumption that this sphere is in contact with all four rollers. Note that the inferred position is used in place of the real object position here, which allows the method to adapt to objects with more extreme aspect ratios. For example, if the object being grasped has a very large aspect ratio (e.g., a long cylinder), the geometric center of the object might not be among the four rollers of the grasper, or even far away from the grasper if the grasper is holding on to one end of a long object [shown in Fig. 4(H)].

The contact location of the object and the roller can subsequently be computed by subtracting the inferred object position and the roller position. The desired motion of the grasped object in combination with the computed contact location would yield a desired contact motion $\delta X_{d,\text{con}}$. This desired contact motion is subsequently projected onto two directions, one of the projected motions is the linear motion resulted from base joint movement, another is the linear motion resulting from the rolling motion

$$\delta X_{\text{contact}} = \alpha \widehat{\delta X}_{cb} + \beta \widehat{\delta X}_{cr}. \tag{2}$$

For all fingers we have

$$\widehat{\delta X}_{cb} = Z_2. \tag{3}$$

By definition, δX_{cr} lies in the contact plane (the plane passing through the contact point and is tangential to both the roller and the object), which can be defined by its normal vector $\hat{n}_{\text{con}}$

The direction of δX_{cr} can then be computed as

$$\widehat{\delta X}_{cr} = \frac{(\delta X_{cb} \times \delta X_{\text{contact}}) \times \hat{n}_{\text{con}}}{\|(\delta X_{cb} \times \delta X_{\text{contact}}) \times \hat{n}_{\text{con}}\|_2}. \tag{4}$$

After solving for the nonorthogonal projection in (4), we have the desired motion for the base joint δX_{cb} and the desired motion for the roller joint δX_{cr}. The position of the pivot joint is then set to be aligned with δX_{cr}. Because the pivot joint has a joint limit of $(-\pi, \pi]$, aligning the pivot joint to δX_{cr} would result in two different pivot joint angles since the roller can roll backward or forward. Out of the two different values, we assign the one that is closer to the current position as the desired pivot joint position for a more efficient operation. [see Fig. 4(F)]

Note that we do not carry out the full inverse kinematics for joint velocities for two reasons. First, the contact locations, in the majority of our grasping configurations, are very close to the axes of the pivot joints, so the fingers are often close to singularities. Second, the pivot joint has a software-enforced joint limit, meaning that the desired instantaneous velocities calculated from the inverse kinematics sometimes could not be reached. The fourth finger, despite having a prismatic base joint, uses the same actuation method as the other three fingers. There is an additional step that maps the desired linear motion of the

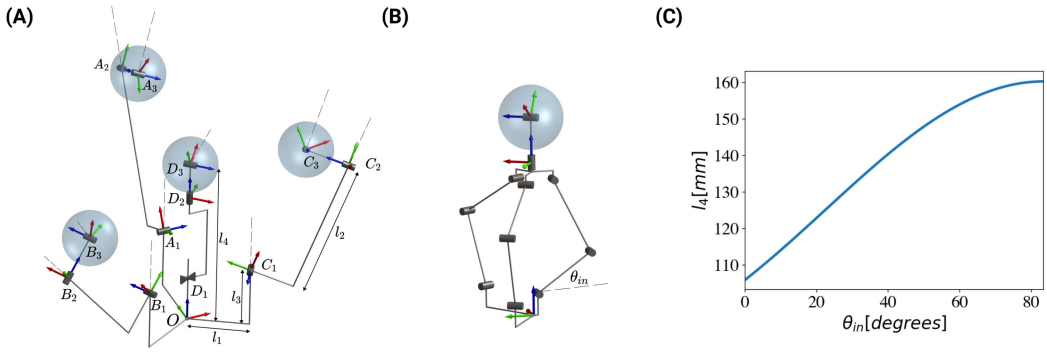


Fig. 5. (A) Reference frame of each of the three joints of each finger, relative to the base coordinate frame. Dashed lines indicate neutral positions from which joint angles are measured. $A_i, B_i, C_i, D_i (i = 1, 2, 3)$ represent local coordinate frames of all joints, X_i, Y_i, Z_i directions of each local coordinate frames are showed as red, green and blue arrows, respectively. (B) Frame representation of the parallel mechanism; (C) Relationship between the input angle and output displacement of the parallel mechanism.

base joint to the revolute joint on the actuator. The conversion from the revolute input to the prismatic output of the parallel mechanism is shown in Fig. 5(C).

V. Experiments

We conducted a quantitative manipulation experiment to evaluate the performance of the manipulation pipeline on the Roller Grasper V3. In addition, we performed a grasping experiment to qualitatively assess the grasping performance of the Roller Grasper V3 on various objects against the gravity.

A. Manipulation Experiment

1) Setup: The manipulation experiment is an ablation study comparing the manipulation framework with and without high-level planning. The experimental setup included the grasper, an overhead RGBD camera (Intel Realsense D430) and various 3-D printed objects including a cube, a rectangular prism, a cube with an opening, a mug, as well as a rectangular prism with a handle (A-E in Fig. 6). These objects were specifically picked to represent a variety of geometrical properties including different aspect ratios, concavities, and irregular protrusions. Object A has an aspect ratio of one and is meant to be an easier to manipulate object. Objects B - E possess some subset of geometrical features including a large aspect ratio, concavities, or protrusions, which makes them more difficult to manipulate. Each object has an embedded Bluetooth sensor (mbientlabs MMC) used to obtain its orientation in real time. There are multiple QR-tags attached to the flat surfaces of the objects that are used for real-time position tracking. While the QR-tags can also provide orientation data, the Bluetooth sensor's orientation output was used because it is much more stable. Combining this data together gives us the object poses. How this pose information is used in our architecture as well as its relationship with the point cloud information in the high-level planner is discussed in detail in Section IV.

We trained individual neural network models for each of the objects A-D, and we also trained a unified model using data collected on all these objects as well as an additional object similar to objects A and D but with a different aspect ratio. The goals of the experiments are as follows: 1) to test whether the grasper's manipulation capabilities deteriorated when using the unified model compared to single models; 2) to verify whether the addition of the high-level planner improved the manipulation results compared to the low-level controller alone; and 3) to test whether the high-level planner could generalize to unseen objects. The experiments compare manipulation results for the conditions listed below with *Condition 3* as a baseline:

Condition 1: Full manipulation architecture for a single model (objects A - D).

Condition 2: Full manipulation architecture for the unified model (objects A - E).

Condition 3: Partial manipulation architecture with only the low-level controller (objects A - E).

There was no data collected using object E; thus only the full manipulation architecture with the unified model and the partial manipulation architecture with only the low-level controller were run using object E.

During each experiment run, the grasper manipulates one of the objects from an initial pose to a target pose, both of which were randomly sampled. Note that the low-level controller attempts to manipulate an object directly from the initial pose to the target pose, without considering stabler waypoints in between. The sampled initial and target positions were limited within a 4cm $\times$ 4cm $\times$ 3cm volume in space to ensure that the objects were graspable. For each tested object, we randomly selected five pairs of starting and target object poses to evaluate the manipulation capabilities of our algorithm. Each pair of starting and target poses was tested on *Conditions 1 - 3* for object A - D. Only *Condition 2* and *Condition 3* were tested on object E. We repeated the experiment five times for each unique pair of starting and target poses and manipulation method (*Conditions 1 - 3*). At the start of each trial, a human operator aligns the superimposed object frame with the specified initial pose shown on the screen. Once aligned, the fingers close and grasp the object. The grasper then manipulates the object until it can not get any closer to the target pose, or the error becomes larger than an empirically determined threshold, which indicates that the object dropped.

We use both the final position error and final orientation error as evaluation metrics for the experiments. We used the

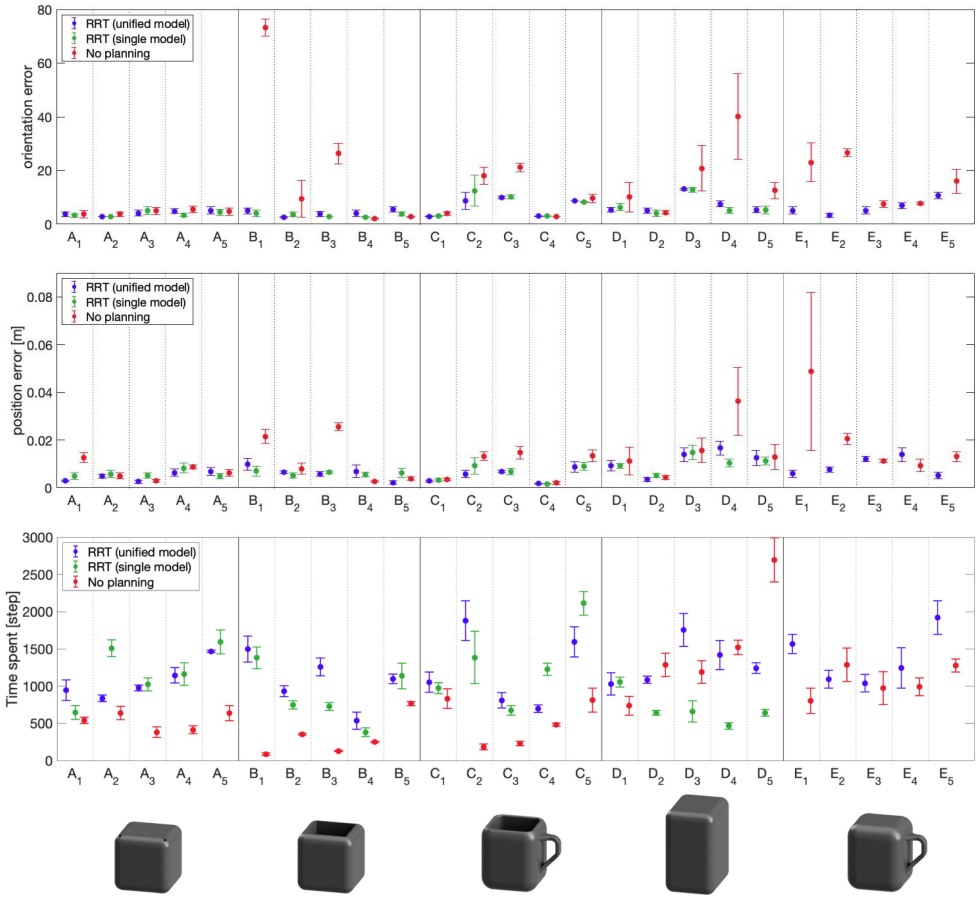


Fig. 6. Experimental results evaluated in orientation error (3) and position error (Euclidean distance) between the final and desired object pose, as well as the time steps taken to complete each case. Each of the five objects (A - E) was tested in five different cases (the subscripts of the object label A - E), and each case was repeated for five times. Object E is the test object without data sampled in simulation.

Euclidean distance between the final and desired object positions as the position error metric, and the quaternion error between the final and desired object orientations suggested in [57] as the orientation error

$$e_\theta = \frac{\min(||q_{obj,d} - q_{obj,c}||_2, ||q_{obj,d} + q_{obj,c}||_2)}{\sqrt{2}}. \quad (5)$$

2) Results: Experimental results from the ablation study are summarized in Fig. 6. For objects with relatively simple geometries (Object A) there was no noticeable performance difference between any of the conditions. However, the high-level planner greatly augments the low-level controller's manipulation capability for objects that are more difficult to manipulate (such as Objects B - E that have concavities, protrusions or larger aspect ratios). Fig. 7 displays selected trajectories of various objects manipulated using the proposed manipulation pipeline.

As shown in Fig. 6, the high-level planner improves the manipulation results in certain situations, most noticeably for objects that have more complex geometries. The improvement is primarily reflected by two aspects. First, the high-level planner allows the grasper to avoid moving the rollers directly into obstacles on the object such as concavities and protrusions, which can cause the rollers to get stuck or the object to be dropped. The improved performance of the high-level planning layer is indicated by a lower final manipulation error. Second, even for cases where the low-level controller alone was able to manipulation objects successfully, the high-level planner was able to find more stable trajectories, which is reflected by narrower error bars in the results. In addition, the result does not show a significant difference between the high-level planner using the unified model versus single model (*Condition 1* vs *Condition 2*). Overall, the full architecture of the manipulation algorithm (*Conditions 1, 2*) demonstrated a great capability to manipulate objects of various shapes in 3-D space, as well as a decent ability to generalize to unseen objects (Object E). Because the high-level planner incorporates intermediate poses, cases using the planner generally require more time to complete. The number of time steps broadly reflects the length of the path taken.

B. Grasping Experiment

While in-hand manipulation can be performed with the palm facing up, we recognized the importance of picking up objects from a surface with the grasper facing downward. In the grasping experiment, the grasper is mounted upside-down on a fixed

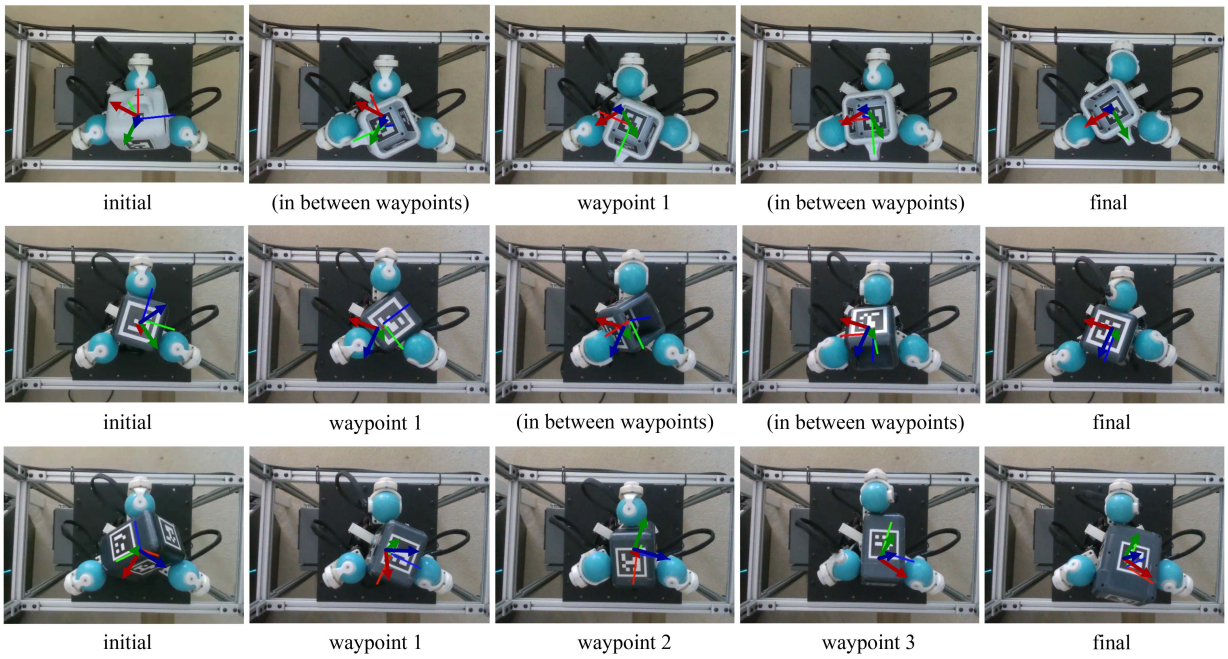


Fig. 7. Trajectories obtained by our manipulation pipeline for different objects. *Top*: Object C (a mug). *Middle*: Object B (a cube with an opening). *Bottom*: Object D (a rectangular prism). The initial and final states, as well as the planned intermediate waypoints, are shown for each trajectory. We also highlight some key frames when manipulating the objects between adjacent waypoints. The current object pose is annotated with thin headless arrows and the planned next pose is annotated with bold arrows in each subfigure.

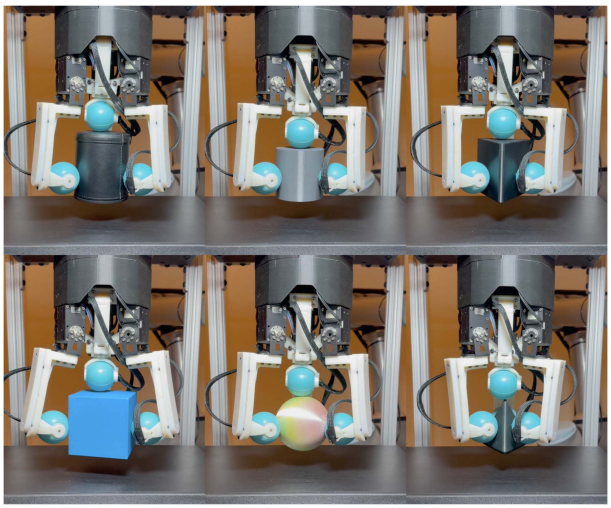


Fig. 8. Grasping objects of various geometries and sizes from a table against gravity. From left to right and top to bottom: a cylinder (synthetic leather), a cylinder (plastic), a triangular prism, a cube, a sphere, and a pyramid.

base to grasp objects resting on a surface below. The grasper closes on the target object and picks it up solely through rolling until the object reaches the roller on the prismatic joint. We tested six objects of different geometries and sizes, including two cylinders, a cube, a prism, a sphere, and a tetrahedron. Each object was grasped five times, and the grasper consistently picked up all the objects without failure, as shown in Fig. 8. These grasping experiments demonstrate how the grasper interacts with objects of various geometries, and show that a stable grasp does not necessarily rely on the support of the palm and can be consistently performed against gravity.

VI. Discussion

A. Assumptions and Limitations

To clarify the scope of this project, there are a few assumptions and limitations that we need to address. First, the Roller Grasper V3 is developed primarily for demonstrating in-hand manipulation capabilities. While the high-friction surfaces of the rollers enable it to pick up objects, pure grasping tasks were not the focus of this work. Investigations on the use of similar rolling contacts for grasping have been presented in [2], [25]. Similarly, while the Roller Grasper V3 is theoretically capable of finger-gaiting, we decided to focus our manipulation strategy on the use of rolling motions (rollers keep in contact with the object).

Second, the low-level controller assumes that the object being grasped is a sphere, which makes it more suitable for objects with low aspect ratios. While this limitation was partially resolved by having the object position inferred by the roller positions (as discussed in Section IV), the underlying assumption still makes the algorithm perform better for object shapes with lower aspect ratios. Theoretically, we could use the actual object geometry in the low-level controller, and subsequently compute accurate analytical solutions. However, planning the path of the contacts on this potentially complex geometry is computationally more expensive and would rely on accurate object pose tracking. Instead, approximating each object by a sphere allows an efficient low-level controller and defers the complexity of in-hand

manipulation planning to the high-level planner that uses a learned function for deciding whether the low-level controller can navigate between two object poses.

Finally, the low-level controller requires access to an object pose estimate relative to the initial object pose. There are a variety of methods for instance or category-level object pose estimation that could be employed (e.g. [58], [59], [60]). However, robust object pose estimation in real time and under potentially severe occlusions during manipulation is challenging, an open research problem and therefore out of the scope of this work. In our experiment, we attached QR-tags on an object's flat surfaces and used embedded Bluetooth sensor data to obtain object poses in real time. This is the primary reason why all objects used in the experiments were cuboidal-shaped. These objects provide a sufficient number of flat surfaces for consistent camera tracking, and the angles between adjacent surfaces are moderate enough to ensure that a new tag is detected promptly by the camera after the previous one becomes occluded during manipulation. Throughout our development, we have qualitatively tested many other types of objects in an open-loop setting, and we have found that objects with more gradual changes in surface curvature are generally easier to manipulate. As object tracking techniques advance, we plan to include a broader range of object types in future research. It is worth noting that for the high-level planner, the point cloud is used as input, allowing the planner to be generalized to different objects. The high-level planner does not require real-time object pose information.

B. Invariant to Object Local Coordinate Definition

When implementing the manipulation architecture, we attach a local coordinate frame to the object and its corresponding point cloud. However, the choice of where this frame is attached to the object is arbitrary and not important. Using this local coordinate frame, a reference pose is defined and all further computation is done using relative transformations with respect to this reference pose. The low-level controller then uses object pose information to transport the object between two consecutive waypoints. To define these relative transformations we need a way to track the object pose, which is accomplished through the instrumentation described above. However, this information could be obtained by any number of alternate methods as previously mentioned.

The high-level planner uses point cloud information to perform reachability checking in each step (i.e., if a goal pose can be reached from a starting pose). The point clouds of the poses to be evaluated are passed into the neural network during planning. This allows the high-level planner to be generalized to different object shapes as we have shown by the manipulation of object E with *Condition 2* in our experiments. Usage of a point cloud also ensures that our reachability checking makes sense physically, since an arbitrary coordinate transformation contains no information about object geometry. Note that the low-level controller does not need to generalize since it always assumes the grasped object is a sphere. To summarize, while we do need to specify an arbitrary object local coordinate frame, neither the high-level planner nor the low-level controller depends on where this coordinate frame is attached to as long as it is consistent over the manipulation sequence.

VII. Conclusion

In this work, we developed the Roller Grasper V3 that is able to perform in-hand manipulation through rolling contacts. This was achieved by steerable spherical rollers located at the fingertips of each finger. We developed a two-stage planner for the grasper consisting of a low-level controller and a high-level planner in order to perform autonomous in-hand manipulation. The low-level controller is a heuristic method that simultaneously controls the pivot joint (Z_2 as shown in Fig. 5), the roller speed, and the base joint speed such that the desired contact velocity is achieved. The high-level planner is a sampling-based path planning method based on the RRT algorithm. When an object is being manipulated from an initial pose to a target pose, the high-level planner finds a feasible path consisting of a sequence of intermediate poses for the object while the low-level controller executes detailed motion in between adjacent poses. We ran an ablation study for the Roller Grasper to manipulate various different objects and showed that the high-level planner indeed improved the manipulation results compared to the cases where only the low-level controller was used. We also showed that the manipulation architecture can generalize to an unseen object.

Acknowledgment

The authors would like to thank Toyota Research Institute provided funds to support this work.

References

[1] S. Leveroni and K. Salisbury, "Reorienting objects with a robot hand using grasp gaits," in *Proc. Robot. Res.*, London: Springer, 1996, pp. 39–51.
[2] 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 *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2020, pp. 9151–9158.
[3] V. Kumar, E. Todorov, and S. Levine, "Optimal control with learned local models: Application to dexterous manipulation," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2016, pp. 378–383.
[4] T. Silver, K. Allen, J. Tenenbaum, and L. Kaelbling, "Residual policy learning," 2018, *arXiv:1812.06298*.
[5] T. Johannink et al., "Residual reinforcement learning for robot control," in *Proc. Int. Conf. Robot. Automat.*, 2019, pp. 6023–6029.
[6] N. Funk et al., "Benchmarking structured policies and policy optimization for real-world dexterous object manipulation," *IEEE Robot. Automat. Lett.*, vol. 7, no. 1, pp. 478–485, Jan. 2022.
[7] T. Li, K. Srinivasan, M.Q.-H. Meng, W. Yuan, and J. Bohg, "Learning hierarchical control for robust in-hand manipulation," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2020, pp. 8855–8862.
[8] C. Piazza, G. Grioli, M. Catalano, and A. Bicchi, "A century of robotic hands," *Annu. Rev. Control Robot. Auton. Syst.*, vol. 2, pp. 1–32, 2019.
[9] M. Diftler et al., "Robonaut 2–The first humanoid robot in space," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2011, pp. 2178–2183.
[10] S. R. Company, "Design of a dextrous hand for advanced CLAWAR applications," in *Proc. 6th Int. Conf. Climbing Walking Robots*, 2003, pp. 17–19.
[11] M. Grebenstein et al., "The DLR hand arm system," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2011, pp. 3175–3182.
[12] W. G. Bircher, A. M. Dollar, and N. Rojas, "A two-fingered robot gripper with large object reorientation range," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2017, pp. 3453–3460.
[13] L. U. Odhner et al., "A compliant, underactuated hand for robust manipulation," *Int. J. Robot. Res.*, vol. 33, no. 5, pp. 736–752, 2014.

[14] L. U. Odhner and A. M. Dollar, "Stable, open-loop precision manipulation with underactuated hands," *Int. J. Robot. Res.*, vol. 34, no. 11, pp. 1347–1360, 2015.
[15] R. R. Ma and A. M. Dollar, "An underactuated hand for efficient finger-gaiting-based dexterous manipulation," in *Proc. IEEE Int. Conf. Robot. Biomimetics*, 2014, pp. 2214–2219.
[16] R. Ma and A. Dollar, "Yale OpenHand project: Optimizing open-source hand designs for ease of fabrication and adoption," *IEEE Robot. Automat. Mag.*, vol. 24, no. 1, pp. 32–40, Mar. 2017.
[17] C. M. McCann and A. M. Dollar, "Design of a stewart platform-inspired dexterous hand for 6-DoF within-hand manipulation," in *2017 IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, 2017, pp. 1158–1163.
[18] H. Stuart, S. Wang, O. Khatib, and M. R. Cutkosky, "The ocean one hands: An adaptive design for robust marine manipulation," *Int. J. Robot. Res.*, vol. 36, no. 2, pp. 150–166, 2017.
[19] J. Borràs and A. M. Dollar, "Dimensional synthesis of three-fingered robot hands for maximal precision manipulation workspace," *Int. J. Robot. Res.*, vol. 34, no. 14, pp. 1731–1746, 2015.
[20] R. R. Ma, N. Rojas, and A. M. Dollar, "Spherical hands: Toward underactuated, in-hand manipulation invariant to object size and grasp location," *J. Mechanisms Robot.*, vol. 8, no. 6, 2016, Art. no. 061021.
[21] P. Datseris and W. Palm, "Principles on the development of mechanical hands which can manipulate objects by means of active control," *J. Mech. Des.*, vol. 107, no. 2, pp. 148–156, 1985.
[22] N. Govindan and A. Thondiyath, "Design and analysis of a multimodal grasper having shape conformity and within-hand manipulation with adjustable contact forces," *J. Mechanisms Robot.*, vol. 11, no. 5, 2019, Art. no. 051012.
[23] V. Tincani et al., "Velvet fingers: A dexterous gripper with active surfaces," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2012, pp. 1257–1263.
[24] R. R. Ma and A. M. Dollar, "In-hand manipulation primitives for a minimal, underactuated gripper with active surfaces," in *Proc. Int. Des. Eng. Tech. Conf.s Comput. Inf. Eng. Conf.*, American Society of Mechanical Engineers, 2016, Art. no. V05AT07A072.
[25] S. Yuan, A. D. Epps, J. B. Nowak, and J. K. Salisbury, "Design of a roller-based dexterous hand for object grasping and within-hand manipulation," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2020, pp. 8870–8876.
[26] J. K. Salisbury and B. Roth, "Kinematic and force analysis of articulated mechanical hands," *J. Mech., Trans., Automat.*, vol. 105, no. 1, pp. 35–41, Mar. 1983.
[27] J. Trinkle, "A quasi-static analysis of dextrous manipulation with sliding and rolling contacts," in *Proc. 1989 Int. Conf. Robot. Autom.*, 1989, pp. 788–793.
[28] F. E. Viña B, Y. Karayiannidis, K. Pauwels, C. Smith, and D. Kragic, "In-hand manipulation using gravity and controlled slip," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2015, pp. 5636–5641.
[29] D. L. Brock, "Enhancing the dexterity of a robot hand using controlled slip," in *Proc. IEEE Int. Conf. Robot. Automat.*, 1988, pp. 249–251.
[30] D. J. Montana, "The kinematics of contact and grasp," *Int. J. Robot. Res.*, vol. 7, no. 3, pp. 17–32, 1988.
[31] V. Kumar, E. Todorov, and S. Levine, "Optimal control with learned local models: Application to dexterous manipulation," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2016, pp. 378–383.
[32] O. M. Andrychowicz et al., "Learning dexterous in-hand manipulation," *Int. J. Robot. Res.*, vol. 39, no. 1, pp. 3–20, 2020.
[33] J. Kerr and B. Roth, "Analysis of multifingered hands," *Int. J. Robot. Res.*, vol. 4, no. 4, pp. 3–17, 1986.
[34] L. Han and J. C. Trinkle, "Dextrous manipulation by rolling and finger gaiting," in *Proc. IEEE Int. Conf. Robot. Automat.*, 1998, pp. 730–735.
[35] A. Simpkins and E. Todorov, "Complex object manipulation with hierarchical optimal control," in *Proc. IEEE Symp. Adaptive Dyn. Program. Reinforcement Learn.*, 2011, pp. 338–345.
[36] C. Chen, P. Culbertson, M. Lepert, M. Schwager, and J. Bohg, "TrajectoTree: Trajectory optimization meets tree search for planning multi-contact dexterous manipulation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2021, pp. 8262–8268.
[37] S. Cruciani, K. Hang, C. Smith, and D. Kragic, "Dual-arm in-hand manipulation and regrasping using dexterous manipulation graphs," 2019, *arXiv:1904.11382*.
[38] M. Murooka, R. Ueda, S. Nozawa, Y. Kakiuchi, K. Okada, and M. Inaba, "Global planning of whole-body manipulation by humanoid robot based on transition graph of object motion and contact switching," *Adv. Robot.*, vol. 31, no. 6, pp. 322–340, 2017.
[39] I. Mordatch, Z. Popović, and E. Todorov, "Contact-invariant optimization for hand manipulation," in *Proc. ACM SIGGRAPH/Eurograph. Symp. Comput. Animation*, 2012, pp. 137–144.
[40] B. Calli and A. M. Dollar, "Robust precision manipulation with simple process models using visual servoing techniques with disturbance rejection," *IEEE Trans. Automat. Sci. Eng.*, vol. 16, no. 1, pp. 406–419, Jan. 2019.
[41] W. G. Bircher, A. S. Morgan, and A. M. Dollar, "Complex manipulation with a simple robotic hand through contact breaking and caging," *Sci. Robot.*, vol. 6, no. 54, 2021, Art. no. eabd2666.
[42] W. G. Bircher, A. S. Morgan, K. Hang, and A. M. Dollar, "Energy gradient-based graphs for planning within-hand caging manipulation," in *Proc. Int. Conf. Robot. Automat.*, 2019, pp. 2462–2467.
[43] R. R. Ma, W. G. Bircher, and A. M. Dollar, "Modeling and evaluation of robust whole-hand caging manipulation," *IEEE Trans. Robot.*, vol. 35, no. 3, pp. 549–563, Jun. 2019.
[44] I. Akkaya et al., "Solving rubik's cube with a robot hand," 2019, *arXiv:1910.07113*.
[45] A. Rajeswaran et al., "Learning complex dexterous manipulation with deep reinforcement learning and demonstrations," 2017, *arXiv:1709.10087*.
[46] H. Van Hoof, T. Hermans, G. Neumann, and J. Peters, "Learning robot in-hand manipulation with tactile features," in *Proc. IEEE-RAS 15th Int. Conf. Humanoid Robots (Humanoids)*, 2015, pp. 121–127.
[47] T. Chen, M. Tippur, S. Wu, V. Kumar, E. Adelson, and P. Agrawal, "Visual dexterity: In-hand dexterous manipulation from depth," 2022, *arXiv:2211.11744*.
[48] H. Zhu, A. Gupta, A. Rajeswaran, S. Levine, and V. Kumar, "Dexterous manipulation with deep reinforcement learning: Efficient, general, and low-cost," in *Proc. Int. Conf. Robot. Automat.*, 2019, pp. 3651–3657.
[49] T. Chen, J. Xu, and P. Agrawal, "A system for general in-hand object re-orientation," in *Proc. 5th Annu. Conf. Robot Learn.*, 2021, pp. 297–307. [Online]. Available: https://openreview.net/forum?id=7uSBJDoP7tY
[50] 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 *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2017, pp. 23–30.
[51] C. Ferrari and J. Canny, "Planning optimal grasps," in *Proc. IEEE Int. Conf. Robot. Automat.*, IEEE Computer Society, 1992, pp. 2290–2291.
[52] F. T. Pokorny and D. Kragic, "Classical grasp quality evaluation: New algorithms and theory," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2013, pp. 3493–3500.
[53] A. Miller and P. Allen, "Examples of 3D grasp quality computations," in *Proc. IEEE Int. Conf. Robot. Automat.*, 1999, pp. 1240–1246.
[54] S. M. LaValle, "Rapidly-exploring random trees: A new tool for path planning," Comput. Sci. Dept., Iowa State Univ., Tech. Rep. 98-11, Oct. 1998.
[55] S. K. Nath, S. Thomas, C. Ekenna, and N. M. Amato, "A multi-directional rapidly exploring random graph (mRRG) for protein folding," in *Proc. ACM Conf. Bioinf. Comput. Biol. Biomed.*, 2012, pp. 44–51.
[56] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "PointNet: Deep learning on point sets for 3D classification and segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 652–660.
[57] D. Q. Huynh, "Metrics for 3D rotations: Comparison and analysis," *J. Math. Imag. Vis.*, vol. 35, no. 2, pp. 155–164, 2009.
[58] M. Wüthrich, P. Pastor, M. Kalakrishnan, J. Bohg, and S. Schaal, "Probabilistic object tracking using a range camera," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2013, pp. 3195–3202.
[59] X. Chen, Z. Dong, J. Song, A. Geiger, and O. Hilliges, "Category level object pose estimation via neural analysis-by-synthesis," in *Proc. Eur. Conf. Comput. Vis.*, Springer, 2020, pp. 139–156.
[60] W. Chen, X. Jia, H. J. Chang, J. Duan, L. Shen, and A. Leonardis, "FS-net: Fast shape-based network for category-level 6D object pose estimation with decoupled rotation mechanism," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 1581–1590.

Shenli Yuan received the dual B.S. degree in mechanical engineering from Purdue University, West Lafayette, IN, USA, and Shanghai Jiao Tong University, Shanghai, China, in 2015, the M.S. degree in mechanical engineering, the M.A. degree in music science and technolog, and the Ph.D. degree in mechanical engineering with a minor in computer science from Stanford University, Stanford, CA, USA, in 2018, 2019, and 2022, respectively.

He is currently a Research Scientist with the Boston Dynamics AI Institute. Prior to this role, he was a Senior Research Engineer at SRI International.

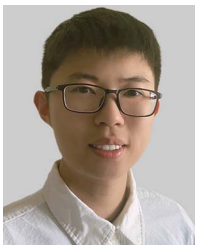
Lin Shao received the B.S. degree from Nanjing University, Nanjing, China, in 2014, the Ph.D. degree from Stanford University, Stanford, CA, USA, in 2021, advised by Jeannette Bohg.

He is currently an Assistant Professor with the Department of Computer Science, National University of Singapore (NUS), School of Computing. His research interests lie at the intersection of robotics and artificial intelligence. His long-term goal is to build general-purpose robotic systems that intelligently perform a diverse range of tasks in a large variety of environments in the physical world. Specifically, his group is interested in developing algorithms and systems to provide robots with the abilities of perception and manipulation.

Dr. Shao is a co-chair of the Technical Committee on Robot Learning in the IEEE Robotics and Automation Society and serves as the Associated Editor at ICRA 2024. His was a recipient of the Best System Paper Award finalist at RSS 2023.

Yunhai Feng received the B.S. degree in computer science and technology from Nanjing University, Nanjing, China, in 2022, and the M.S. degree in computer science from the University of California, San Diego, CA, USA, in 2024. He is currently working toward the Ph.D. degree in computer science with Cornell University, Ithaca, NY, USA.

He also spent time as a Research Intern with the Boston Dynamics AI Institute, Cambridge, MA, USA. His research is focused on data-driven and model-based decision-making.



Jiatong Sun received the bachelor of engineering degree majoring in mechanical engineering from Shanghai Jiao Tong University, Shanghai, China, in 2020, and the master of science degree in engineering from the University of Pennsylvania, in 2022, majoring in mechanical engineering and computer science.

He is currently a software engineer with WeRide.ai, specializing in motion planning algorithms for L4 autonomous driving. . He has contributed to various projects in autonomous robotics, aiming to advance the field through innovative research and practical applications. His research focuses on learning-based robot manipulation and autonomous racing car trajectory optimization.

Teng Xue (Graduate Student Member, IEEE) received the M.Sc. degree in mechanical engineering from Shanghai Jiao Tong University, Shanghai, China, in 2020. He is currently working toward the Ph.D. degree in electrical engineering with École Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland.

He is also a Research Assistant with the Robot Learning and Interaction Group at Idiap Research Institute, Martigny, Switzerland. Previously, he was a visiting student with the Robotic Systems Lab, ETH Zurich, Zurich, Switzerland. He is broadly interested in planning and control for robotic systems that can physically interact with the world, with a specific focus on task and motion planning, contact-rich manipulation, and learning from demonstration.

Connor L. Yako received the B.S. degree in mechanical engineering from the University of Michigan, Ann Arbor, MI, USA, in 2018, the M.S. degree in mechanical engineering from Stanford University, Stanford, CA, USA, in 2020. He is currently working toward the Ph.D. degree in mechanical engineering with Stanford University.

His research is focused on nonanthropomorphic methods for performing in-hand manipulation, such as a gripper that changes its finger dimensions as well as asymmetric vibrations.

Jeannette Bohg (Member, IEEE) received the Diploma in computer science from the Technical University in Dresden, Dresden, Germany, in 2005, the master degree in art and technology from the Chalmers University, Gothenburg, Sweden, in 2007, and the Ph.D. degree from the Division of Robotics, Perception and Learning (RPL), KTH Stockholm, Stockholm, Sweden, in 2011.

She is currently an Assistant Professor of computer science with Stanford University. She was a Group Leader with the Autonomous Motion Department (AMD) of the MPI for Intelligent Systems until September 2017. Her research focuses on perception and learning for autonomous robotic manipulation and grasping.

Dr. Bohg has received several Early Career and Best Paper awards, most notably the 2019 IEEE Robotics and Automation Society Early Career Award, the 2020 Robotics: Science and Systems Early Career Award and the 2023 Sloan Fellowship.

J. Kenneth Salisbury (Life Member, IEEE) received the M.S., M.A., and Ph.D. degrees in electrical and mechanical engineering from Stanford University, Stanford, CA, USA, in 1975, 1977, and 1982, respectively.

Since then he has held Research Professorships with MIT and Stanford. During a four-year sabbatical at Intuitive Surgical he helped develop the da Vinci surgical robot. He has been associated with many fundamental contributions in robotics and haptics, including the PHANToM Haptic Interface, the MIT WAM Arm, and the Willow Garage PR-2. He is currently a Professor Emeritus with Stanford in the Computer Science and Surgery Departments, conducting research in in-hand manipulation, physical Human/Robot Interaction (pHRI) (in a robotic EMT), and patient-specific simulation of skull-based surgical procedures.