

Learning to Use Chopsticks in Diverse Gripping Styles

ZESHI YANG, Simon Fraser University, Canada and CFCS Peking University, China

KANGKANG YIN, Simon Fraser University, Canada

LIBIN LIU*, CFCS Peking University, China

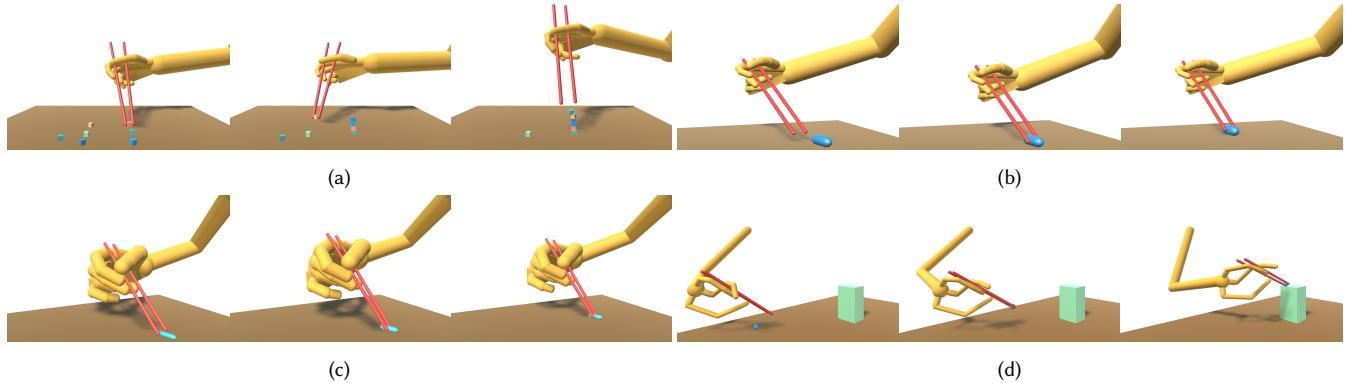


Fig. 1. Our system learns how to use chopsticks in diverse gripping styles for multiple hand morphologies. The trained physics-based hand controllers can pick up and relocate objects of various shapes and sizes in realtime.

Learning dexterous manipulation skills is a long-standing challenge in computer graphics and robotics, especially when the task involves complex and delicate interactions between the hands, tools and objects. In this paper, we focus on chopsticks-based object relocation tasks, which are common yet demanding. The key to successful chopsticks skills is steady gripping of the sticks that also supports delicate maneuvers. We automatically discover physically valid chopsticks holding poses by Bayesian Optimization (BO) and Deep Reinforcement Learning (DRL), which works for multiple gripping styles and hand morphologies without the need of example data. Given as input the discovered gripping poses and desired objects to be moved, we build physics-based hand controllers to accomplish relocation tasks in two stages. First, kinematic trajectories are synthesized for the chopsticks and hand in a motion planning stage. The key components of our motion planner include a grasping model to select suitable chopsticks configurations for grasping the object, and a trajectory optimization module to generate collision-free chopsticks trajectories. Then we train physics-based hand controllers through DRL again to track the desired kinematic trajectories produced by the motion planner. We demonstrate the capabilities of our framework by relocating objects of various shapes and sizes, in diverse gripping styles and holding positions for multiple hand morphologies. Our

*corresponding author

Authors' addresses: Zeshi Yang, Simon Fraser University, Vancouver, Canada, and CFCS Peking University, Beijing, China, zeshiy@sfu.ca; Kangkang Yin, Simon Fraser University, Vancouver, Canada, kkyin@sfu.ca; Libin Liu, CFCS Peking University, Beijing, China, libin.liu@pku.edu.cn.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

© 2022 Association for Computing Machinery.

0730-0301/2022/7-ART95 \$15.00

<https://doi.org/10.1145/3528223.3530057>

system achieves faster learning speed and better control robustness, when compared to vanilla systems that attempt to learn chopstick-based skills without a gripping pose optimization module and/or without a kinematic motion planner. Our code and models are available at this link.¹

CCS Concepts: • Computing methodologies → Animation; Physical simulation; Motion planning; • Theory of computation → Reinforcement learning.

Additional Key Words and Phrases: Physics-based Character Animation, Motion Synthesis, Deep Reinforcement Learning, Bayesian Optimization, Manipulation, Grasping, Tool Use

1 INTRODUCTION

Dexterous manipulation and tool usage has been a long-standing challenge in computer animation and robotics. The main difficulties of tool use include the high degrees of freedom of the hands; the underactuation of the tools; and the complex interplay between the hands, tools and objects. The difficulty level also depends on the type of tools involved. Some tools only need to be grasped firmly in hand, such as hammers. Some tools need to be grasped and manipulated by hand, such as scissors. In this paper, we consider one of the most challenging tools: chopsticks.

Chopsticks are pairs of equal-length sticks used in Asian dining for millennia. Their simple design, or lack of design, poses several challenges in terms of control. First, the hand needs to grip and manipulate two independent sticks at the same time. Second, there are no obvious holding structures on the chopsticks, such as finger rings for a pair of scissors, to stabilize hand-tool contacts. Lastly, the chopstick-object contacts lie at the tip of the chopsticks, which are usually far away from the chopstick-hand contact points near the rear end of the chopsticks. Children usually need years of practice

¹<https://github.com/chopsticks-research2022/learning2usechopsticks>

to master chopstick maneuvers. Even adults may find learning chopsticks challenging, if they did not grow up using them. The steep learning curve associated with chopsticks usage has spurred many video tutorials on YouTube, and the invention of training chopsticks that provide position retainer loops to stabilize finger-stick contacts.

The simple form of chopsticks, however, does enable their popularity and versatility. An estimated 33% of the world’s population use chopsticks on a daily basis. Chopsticks are available everywhere, on dining tables as well as in the woods. They can pick up and move all kinds of foods: rice, meat, or noodles. They can manipulate in many different ways, so that spatulas, whisks, or pasta ladles are not necessary in Asian cooking. In robotics, research has been carried out to adopt chopsticks for eating assistance [Chang et al. 2007; Yamazaki and Masuda 2012], micro-manipulation [Ramadan et al. 2009], and medical surgery [Joseph et al. 2010; Ragupathi et al. 2010; Sakurai et al. 2016].

The practicality and generality of chopsticks come at the cost of control complexity. In robotics, the chopsticks are usually rigidly attached to robot arms with reduced Degrees of Freedom (DoFs). In graphics, research on using chopsticks is nonexistent so far, to the best of our knowledge. Chopsticks usage is emblematic of a wider category of difficult-to-solve multi-contact manipulation-and-control problems. We focus on solving the problem of using truly underactuated chopsticks, with reasonable robustness in diverse gripping styles and holding positions for multiple morphologies. Inspired by how parents teach children chopsticks skills, we tackle this challenging control problem by decomposing it into two subproblems: how to hold chopsticks properly? And then how to manipulate objects using chopsticks?

To use chopsticks effectively, users need to firstly hold them firmly. Although there is a consensus on a so-called standard grip being the most efficient way to use chopsticks [Yamauchi et al. 2010], many people find their own ways to grip chopsticks during learning [Mukai and Hashimoto 1978; Osera et al. 2018; Yamakawa et al. 2018], such as the various grips shown in Figure 2. We characterize a gripping style by the contact relationships between each finger and either chopstick. We optimize the gripping pose in a particular style by combining deep reinforcement learning and Bayesian optimization. Such an approach enables automatic discovery of diverse gripping poses for unusual hand morphologies. Using the output grips of our BO optimization, a moving virtual hand can hold the chopsticks firmly in physics simulation, and achieve some basic open-and-close chopsticks maneuvers.

To use chopsticks proficiently, users also need to control finger movements precisely in order to relocate objects via the tips of the chopsticks. Such fine motor controls are most likely impossible to design manually, which were possible for locomotion tasks. We design a two-level control system that first plans the chopsticks movements kinematically, and then trains physics-based hand controllers via model-free deep reinforcement learning. The high-level kinematic motion planner consists of a grasping model to select the best chopsticks configuration for grasping the objects, and then a trajectory generator to optimize for a collision-free chopsticks trajectory based on the start and goal transformations of the object. The hand and arm trajectories are then solved from the desired chopsticks trajectory using inverse kinematics. All the planned reference

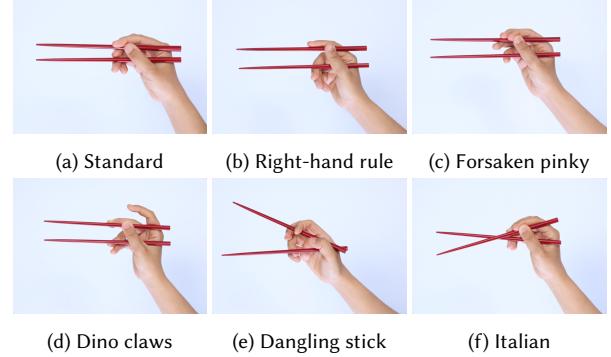


Fig. 2. Multiple ways of holding chopsticks. Our system can discover similar gripping poses for seventeen styles, many of which correspond to the commonly used chopstick grips given in [Macro 2021].

trajectories along with the optimized gripping pose in a desired style are then passed to the DRL system to train the low-level hand controls using simple tracking rewards. Our learned low-level hand controllers are able to grasp and move or throw objects of various shapes and sizes in realtime, and the high-level motion planner can plan or replan trajectories at interactive rates.

In summary, the contributions of this work are mainly twofold:

- (1) We present a learning and control framework for object relocation using chopsticks. The high-level motion planner synthesizes collision-free kinematic trajectories at interactive rates, and the low-level physics-based hand controllers track the planned trajectories in realtime once trained. No sophisticated reward tuning or motion capture of human demonstrations are needed.
- (2) We use Bayesian optimization combined with deep reinforcement learning to discover physically valid gripping poses in multiple styles. The optimized grips correspond well to human experiences and no manual specification is needed. Such an imitation-free method can generalize easily to different hand morphologies and is thus applicable to a broad range of graphics and robotics applications.

2 RELATED WORK

Human hands and tool usage are the special anatomy and function that helped driving the evolution of human brain, which ultimately differentiated humans from the rest of the animal kingdom. Significant amount of research endeavors have been invested into tackling the challenging problem of synthesizing dexterous manipulation in simulation or on robots. We classify the most relevant recent works into two categories: hand manipulation and tool usage. Hand manipulation is manipulation of objects directly by fingers and the palm, such as grasping and relocation of objects [Kry and Pai 2006; Liu 2009; Zhao et al. 2013], and in-hand manipulation [Zhang et al. 2021]. Tool usage is manipulation of objects by tools that are operated by hands or robot arms. We refer interested readers to other orthogonal dimensions of research in hand and finger animation to the excellent survey provided in [Wheatland et al. 2015].

2.1 Hand Manipulation

2.1.1 Kinematic Methods. Hand manipulations such as grasping or playing musical instruments can be synthesized through traditional inverse kinematic methods [Aydin and Nakajima 1999; ElKoura and Singh 2003; Huang et al. 1995; Kim et al. 2000; Koga et al. 1994]. More recently, deep neural networks have also been utilized to synthesize hand manipulations [Karunratnakul et al. 2020; Taheri et al. 2020]. For example, [Taheri et al. 2020] constructed a hand-object interaction database through motion capture, and trained a neural network to predict object grasping poses for human hands. [Zhang et al. 2021] showcased impressive hand-object manipulation results such as turning a torus in hands. Kinematic methods, however, cannot generate motions responsive to a dynamic environment. The synthesized manipulations could also display artifacts such as penetrations into the objects. Moreover, data-driven kinematic methods usually require high-fidelity hand manipulation capture, which is hard and expensive in most cases.

2.1.2 Physics-based Methods. Physics-based control methods leverage physics simulation to generate motions with physical realism and environmental interactions. The key challenge, however, is to design or learn robust controllers to drive the simulated characters or hands. High-fidelity motion capture data has been utilized to help with synthesizing physically realistic hand manipulations [Kry and Pai 2006; Zhao et al. 2013]. In the absence of motion capture data, trajectory optimization provides a viable approach to synthesizing physics-based dexterous manipulations [Liu 2008, 2009; Mordatch et al. 2012; Wang et al. 2013; Ye and Liu 2012]. These methods usually need to model the dynamics to a great extent, such as incorporating friction constraints into the objective functions to avoid undesirable movements between hands and objects during grasping. Therefore, such methods usually simplify the dynamics to a certain degree to reduce the control complexity. For our case of dealing with five fingers and two chopsticks, the contact and friction dynamics will probably be too overwhelming to handle, even with simplified physics.

We therefore opt for a model-free approach using deep reinforcement learning. DRL has been widely used in computer graphics and robotics to learn diverse motion skills, such as locomotion [Bergamin et al. 2019; Park et al. 2019; Peng et al. 2018a, 2017, 2021; Tan et al. 2018], athletic skills [Liu and Hodgins 2017; Yin et al. 2021], and manipulations [Andrews and Kry 2013; Chen et al. 2021; Garcia-Hernando et al. 2020; Nagabandi et al. 2020; Popov et al. 2017; Rajeswaran et al. 2018]. Challenging manipulation tasks, such as solving a Rubik’s cube with a robot hand, have been demonstrated using DRL-based control learning methods [Akkaya et al. 2019]. It remains an open problem how to design suitable DRL reward functions to learn natural-looking skills for complex tasks, however. One idea is to leverage human demonstrations as reference skills for physical agents to imitate, which has been proven to improve both the learning efficiency and control robustness [Rajeswaran et al. 2018]. Multiple reference skills can be combined to help solve more challenging tasks using hierarchical deep reinforcement learning, such as dribbling a soccer ball or carrying objects to target locations [Merel et al. 2020; Peng et al. 2019]. More recently, adversarial

imitation learning DRL system has been quite successful at learning motion priors from large datasets of unstructured motion clips [Peng et al. 2021]. The motion priors obviate the need for manually designed imitation objectives or a high-level motion planner.

Capturing chopsticks skills, however, may be impractical as severe occlusions and subtle movements are involved. We also wish our solution to generalize well to graphics applications with monster-hand morphologies, and robotics applications with non human-hand-like manipulators. We thus rely on Bayesian optimization coupled with DRL to discover diverse and physically valid chopsticks gripping poses. We then use a motion planner to generate chopsticks configurations and trajectories to satisfy kinematic task objectives. The discovered gripping poses and synthesized trajectories are then passed to our DRL-based training system to learn chopsticks skills using simple tracking rewards. Therefore, the advantages of our system include its simplicity in terms of system setup and tuning, and diversity in terms of gripping styles and hand morphologies.

2.2 Tool Usage

Tool usage has not been explored too much in the graphics community. The most relevant work is [Zhang et al. 2020], which employed DRL to learn control policies to manipulate amorphous materials, such as gathering rice with scrapers or flipping pancakes with pans. Their tools are driven by a virtual proportional derivative controller. Then hand motions are reconstructed via inverse kinematics. There are more research activities on tool usage in the robotics community [Fang et al. 2020b; Ke et al. 2020, 2021; Kim et al. 2021; Toussaint et al. 2018; Wu et al. 2019], although most of them turn it into a simpler problem by attaching the tools directly onto the robot arm. For example, chopsticks are attached to a robot arm to grasp objects in [Ke et al. 2021], where human demonstrations were also used to help the learning of grasping policies. We study the problem of controlling underactuated chopsticks by hands, meaning that the chopsticks can actually move inside and fall out of the hand. We have not been able to find any prior work on exactly the same problem in the literature.

2.3 Bayesian Optimization

Bayesian Optimization (BO) is a class of optimization methods for expensive black-box function optimization. The function is optimized purely through evaluations as no gradient information is readily available. In Bayesian optimization, a Bayesian statistical model, such as a Gaussian Process [Rasmussen 2003], is maintained to predict the values and uncertainties of the objective function. An acquisition function is applied to query the most promising and informative regions based on current estimations. Recently BO has seen more and more adoptions in robotics and computer graphics for various applications [Brochu et al. 2010, 2007; Hu et al. 2021; Koyama et al. 2020]. Most relevant to our work, BO has been employed to tune parameters of a bipedal locomotion controller [Rai et al. 2018], optimize hyperparameters of physics-based character animation systems [Yang and Yin 2021], and discover diverse athletic jumping strategies [Yin et al. 2021]. For our problem, the chopsticks gripping pose can be viewed as a hyperparameter of the hand control policies trained through DRL, and therefore optimized by BO. There

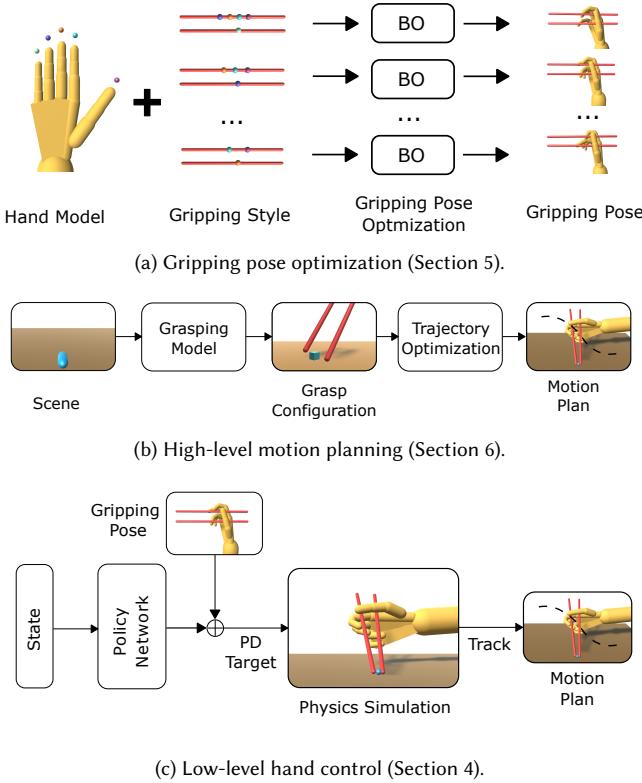


Fig. 3. The three main components of our learning and control framework. (a) For a desired gripping style, we employ BO and DRL to optimize for a physically valid gripping pose of the hand. (b) To achieve an object relocation task, the motion planner first selects a suitable chopsticks configuration for grasping, and then synthesizes collision-free trajectories for the chopsticks and hand. (c) Then we train policy networks using DRL to track the planned trajectories for a chosen gripping pose.

are multiple BO algorithms, such as Gaussian-Upper Confidence Bound (GP-UCB) [Srinivas et al. 2010] and Entropy Search [Hennig and Schuler 2012]. Our gripping pose optimization is based on the GP-UCB implemented with the Gaussian process framework [GPy 2012]. Our results show that the optimized poses correspond well to commonly used chopstick grips by humans.

3 OVERVIEW

Figure 3 provides an overview of our learning and control framework. Taken the models of one hand and two chopsticks as input, our goal is to learn robust hand controls to use the chopsticks for a variety of object grasping and relocation tasks. We achieve this goal by solving two sub-problems: first finding physically valid gripping poses in multiple styles in a *gripping pose optimization* step, and then learning *hand control policies* to hold the chopsticks in a specific gripping pose to pick up and relocate objects in simulation.

In the gripping pose optimization step as illustrated in Figure 3a, we employ Bayesian Optimization (BO) to find the optimal gripping pose for a desired style. A gripping style is characterized by a set of finger-chopstick contact relations, or specifically, which

finger should be in contact with which stick. Gripping styles can be specified either manually or automatically as will be described later. Given a desired gripping style, our BO algorithm iteratively proposes a set of finger-stick contact positions, which are then converted into a hand pose using Inverse Kinematics (IK). We then employ DRL to evaluate the quality of the candidate gripping pose by training policies to perform simple open-and-close chopsticks maneuvers. The performance of the trained policy, characterized by its average reward in simulation, is then fed back to the BO to come up with the next proposal for contact positions.

After good gripping poses are found, we build hand controllers to relocate objects with chopsticks for each gripping pose. To this end, we design a two-level learning and control framework. The high-level kinematic motion planner as shown in Figure 3b first selects the best chopsticks configuration in order to grasp the object of interest. We develop a neural-network based grasping model to recommend such chopsticks configurations based on the shape and transformation of the object. The planner then optimizes for collision-free trajectories for the chopsticks and hand, based on the relocation task required. Next the low-level hand control policy as shown in Figure 3c is trained through model-free DRL to track the generated motion plan. Once trained, the control policies can track novel reference trajectories generated by the planner in realtime to relocate objects of various shapes and sizes.

4 PHYSICS-BASED HAND TRACKING CONTROL

We train tracking controllers for the hand to follow given trajectories of the object, the chopsticks, the hand, and the arm, while holding the chopsticks in a specific gripping pose. The trajectories to track are generated by a high-level motion planner in our system, but motion capture examples or manually defined keyframe animations could be used here if so wished. We use Deep Reinforcement Learning (DRL) to train these tracking controllers. The same DRL components are used for both the gripping pose optimization and the final control policy learning for actual object manipulations, although their reward terms are slightly different.

4.1 Deep Reinforcement Learning

Our policy learning is formulated as a standard Reinforcement Learning (RL) problem, where an agent interacts with the environment and learns from its experience for the optimal control policy that maximizes the total reward. We denote the control policy as $\pi_\theta(a_t|s_t)$, which models the conditional distribution of the action a_t given the current state s_t , and θ represents the learnable parameters. At each time step t , the agent takes an action a_t according to π_θ , gets a reward r_t from the environment, and then transits to another state s_{t+1} according to a probabilistic dynamics model $p(s_{t+1}|s_t, a_t)$. Starting from an initial state s_0 , this procedure will generate a trajectory $\tau = \{s_0, a_0, s_1, a_1, \dots\}$. We calculate the cumulative reward of τ with a discount factor γ as $R(\tau) = \sum_t \gamma^t r_t$. Then the expected return $J(\theta)$ is computed over all possible trajectories induced by policy π_θ and the dynamics p as:

$$J(\theta) = \mathbb{E}_{\tau \sim p_{\theta(\tau)}} [R(\tau)] \quad (1)$$

where $p_{\theta(\tau)} = p(s_0) \prod_t \pi_\theta(a_t|s_t) p(s_{t+1}|s_t, a_t)$, and $p(s_0)$ is the initial state distribution.

We train our policies using the Proximal Policy Optimization (PPO) algorithm [Schulman et al. 2017], which has been frequently adopted in physics-based character animation and robotics for its robustness and simplicity. PPO is often implemented within an actor-critic framework, where a critic network is trained to estimate the value of states, and an actor network is trained to output the control policy π . We employ multi-step returns $TD(\lambda)$ and the generalized advantage estimator $GAE(\lambda)$ [Schulman et al. 2015] to facilitate training of the neural networks, similar to the PPO implementation in [Peng et al. 2018a] where interested readers can find more details.

4.2 Simulation Setup

We simulate our hand models of potentially different morphologies together with a common two-link arm model as an articulated rigid-body system, allowing a moderate task space for object relocation tasks. The shoulder joint is fixed in position and only has three rotational Degrees of Freedom (DoFs). We note that hereafter whenever we refer to the state or pose of the hand, we really mean the state or pose of the whole hand-and-arm structure. We parameterize the state of the hand in generalized coordinates as $s_{\text{hand}} = (\mathbf{q}, \dot{\mathbf{q}})$, where \mathbf{q} represents the joint angles and $\dot{\mathbf{q}}$ the rotational speeds. All joints are actuated using PD-servos, and the positional PD targets $\hat{\mathbf{q}}$ are computed by the control policy π . The joint torques τ are then computed as

$$\tau = k_p(\hat{\mathbf{q}} - \mathbf{q}) - k_d\dot{\mathbf{q}} \quad (2)$$

where k_p and k_d are the stiffness and damping parameters of the joint-level PD-servos.

We model chopsticks as a pair of long rigid capsules of the same length and radius. The two sticks can move independently so the total DoFs of a pair of chopsticks is twelve. We denote the state of the chopsticks as $s_{\text{chop}} = (\mathbf{p}_i, \mathbf{o}_i, \mathbf{v}_i, \omega_i), i = 1, 2$, where \mathbf{p}_i is the position of the Center of Mass (CoM) of Chopstick i , \mathbf{o}_i , \mathbf{v}_i and ω_i are the orientation, linear velocity and angular velocity of Chopstick i . Unless otherwise noted, the rotations are parameterized in quaternions in our system. The upper stick is indexed as Chopstick 1 and the lower stick is Chopstick 2, as shown in Figure 4.

The objects being manipulated by the chopsticks are modeled as rigid bodies from a predefined set of shapes and range of sizes as shown in Table 1. We use a tuple t_{obj} to indicate the shape and size of the corresponding object. The state of an object is denoted by a tuple $s_{\text{obj}} = (\mathbf{p}_{\text{obj}}, \mathbf{o}_{\text{obj}}, \mathbf{v}_{\text{obj}}, \omega_{\text{obj}})$, which consists of the object's position, orientation, and linear and angular velocity.

4.3 Learning of Tracking Control

We model our hand tracking control policy π as a fully-connected neural network with two 256-unit hidden layers and ReLU activations. We train these controllers using the PPO algorithm as described in Section 4.1, where the value network shares the same structure as the policy network except that its last layer is a single linear unit. At runtime, the controller π is responsible for computing an action a given the current state s of the system.

4.3.1 States and Actions. The state s is the input to the hand controllers. In our system, s consists of the simulation states of the chopsticks, the hand, and the object being manipulated, as well as the contact information between them. We compute the state of

the chopsticks and the object with respect to the local coordinate frame of the palm, to facilitate learning and improve robustness. In addition, a short segment of the desired motion trajectory is also included as part of s to facilitate tracking. More specifically, the desired kinematic states of the hand, chopsticks, and objects of the next six frames from the planned trajectories are included in s . A complete list of all the components of s is provided in Appendix A.

The action a is the output of the hand controllers. In our system, a represents a corrective offset pose δq that will be added to a chosen gripping pose q^* to compose the final target pose \hat{q} for the PD-servos as in Equation 2.

4.3.2 Rewards. The reward function is designed to encourage the hand to hold the chopsticks firmly in a chosen style and move the object following a desired trajectory. More specifically, it consists of four reward terms:

$$r = e^{r_{\text{hand}} + r_{\text{chop}} + r_{\text{object}} + r_{\text{contact}}} \quad (3)$$

The hand control term r_{hand} encourages the hand and arm to match their planned trajectories:

$$r_{\text{hand}} = -10 \|\mathbf{q}_{\text{hand}} - \tilde{\mathbf{q}}_{\text{hand}}\| \quad (4)$$

where \mathbf{q}_{hand} is the simulated hand pose, and $\tilde{\mathbf{q}}_{\text{hand}}$ is the desired hand pose in the planned trajectory.

Similarly, the chopsticks term r_{chop} and the object term r_{obj} are defined as:

$$r_{\text{chop}} = -40 \sum_{i \in \{1, 2\}} \|\mathbf{p}_i - \tilde{\mathbf{p}}_i\| - 10 \sum_{i \in \{1, 2\}} \Theta(\mathbf{o}_i, \tilde{\mathbf{o}}_i) \quad (5)$$

$$r_{\text{obj}} = -40 \|\mathbf{p}_{\text{obj}} - \tilde{\mathbf{p}}_{\text{obj}}\| - 10 \cdot \Theta(\mathbf{o}_{\text{obj}}, \tilde{\mathbf{o}}_{\text{obj}}) \quad (6)$$

where \mathbf{p} and \mathbf{o} represent positions and orientations respectively. The scalar function $\Theta(\mathbf{o}_1, \mathbf{o}_2)$ computes the absolute angle between two quaternions \mathbf{o}_1 and \mathbf{o}_2 .

Lastly, the term r_{contact} prevents fingertips from leaving or slipping on the chopsticks:

$$r_{\text{contact}} = -10 \cdot \sum_{i=1}^{i=N} d_i \quad (7)$$

where d_i is the minimal distance between the fingertip i (the part called distal phalanx in hand anatomy) and its desired contact position on the chopsticks as determined by the gripping style. N is the number of fingers that should remain in contact with the chopsticks in the corresponding gripping style, as will be described next.

5 GRIPPING POSE OPTIMIZATION

A good chopstick gripping pose is a strong determinant of successful manipulations using chopsticks. Some gripping poses make it easy and efficient to use chopsticks, while others may feel awkward or even make it infeasible to manipulate with chopsticks. The goal of the gripping pose optimization component is to find the optimal pose with which the hand can hold the chopsticks steadily while still can move in a coordinated and flexible fashion to accomplish object manipulation using the tips of the chopsticks. Such an objective is difficult to formulate in closed form, thus we opt for a learning-based evaluation scheme coupled with Bayesian optimization.

Algorithm 1: Gripping Pose Optimization with BO and DRL

Input: A gripping style c ; maximal BO iterations $maxIter$.
Output: The optimal gripping pose \mathbf{q}^* for c .

```

1  $i = 0; r^* = -\inf$ 
2 while  $i < maxIter$  do //BO iterations
3    $x \leftarrow$  contact position proposal by BO from  $c$ ;
4    $\mathbf{q} \leftarrow$  solving gripping pose by IK from  $x$ ;
5    $r \leftarrow$  average reward of DRL-trained policy for  $\mathbf{q}$ ;
6   update BO record with  $x, r$ ;
7   update Gaussian surrogate model;
8   if  $r > r^*$  then  $(\mathbf{q}^*, r^*) \leftarrow (\mathbf{q}, r)$ ;
9    $i++$ ;
10 end
11 return  $\mathbf{q}^*$ 
```

More specifically, we learn a control policy using reinforcement learning to track basic chopsticks maneuvers for a candidate gripping pose. The performance of the learned policy is then used as an assessment of the gripping pose. As the DRL-based evaluation is relatively expensive and not differentiable, we employ the sample-efficient and gradient-free Bayesian optimization to suggest candidate poses. Our gripping pose optimization algorithm is summarized in Algorithm 1. Note that the arm is controlled to maintain a static pose at this stage, and dynamic arm movements will be synthesized later in Section 6.

5.1 Bayesian Optimization

Bayesian Optimization (BO) is one of the ideal choices for optimizing expensive black-box functions with no gradients. It is designed to minimize the number of function evaluations by querying the most promising and informative data points. For a given objective function, BO searches for its optimum through a series of evaluations, where the history of those evaluations are recorded to fit an *acquisition function*, which will determine the next candidate data point. Our gripping pose optimization is based on the GP-UCB algorithm [Srinivas et al. 2010] implemented with the Gaussian process framework [GPy 2012]. Interested readers can refer [Srinivas et al. 2010] for more details. We set the maximal allowed number of function evaluations to ten in our implementation.

5.2 Chopsticks Gripping Style

In our system, a chopsticks gripping style is characterized by the contact relationships between each finger and either chopstick. For a human hand with five fingers, namely the thumb, index, middle, ring, and little finger, we can represent a gripping style with a 5-tuple $c = (c_1, c_2, c_3, c_4, c_5)$, where $c_i = j, j \in \{0, 1, 2\}$ indicates that finger i should be in contact with chopstick j . 0 denotes no contact. Following this notation, the standard gripping style shown in Figure 2 can be represented by the tuple $(1, 1, 1, 2, 0)$, for example. In general, for a hand model with N fingers, its chopsticks gripping style can be defined with an N -tuple.

For hands with a small number of fingers, we could simply enumerate all values of the gripping style tuple and optimize a gripping

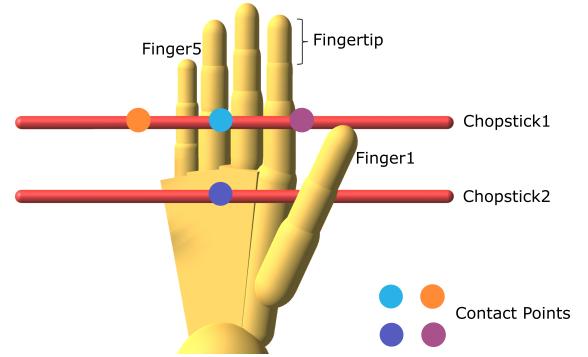


Fig. 4. The default T-Pose used by our IK solver. The upper stick is indexed as Chopstick 1 and the lower stick is Chopstick 2. Fingers are indexed from the thumb to pinky as Finger 1 to 5. A fingertip is the first segment of a finger.

pose for each of them. A human hand, for example, has potentially a maximal of $3^5 = 243$ gripping styles. However, many of them may be infeasible, inefficient, or unnatural. We thus employ two heuristics to prune the style space to eliminate bad styles, as well as to reduce the optimization workload. First, the thumb plays a crucial role in performing precise tool use, and therefore we do not allow c_1 to be zero. We also observe that the thumb is usually used to support the upper Chopstick 1 as shown in Figure 2. This is because the lower stick has some default support from the valley between the thumb and the index finger, while the upper stick needs the thumb more for support and movement. Consequently, we set $c_1 = 1$ and shrink the style space by $2/3$. Second, finger crossing usually leads to awkward or infeasible grasping poses, thus the gripping styles with finger crossings are excluded from further pose optimization. After applying these two heuristics, there are seventeen gripping styles left, for each of which we run BO to obtain an optimized gripping pose.

5.3 Gripping Pose Generation

When given a chopsticks gripping style as input, represented by an N -tuple, we use BO to search for the optimal gripping pose for that style. To reduce the degrees of freedom of the problem, we first optimize the finger-stick contact positions according to the contact patterns specified in the style tuple. Specifically, we parameterize each contact position using a single scalar x representing the contact location along the chopstick. Then the BO algorithm just needs to optimize a vector \mathbf{x} up to N dimensions. Once the contact positions are determined, the gripping pose can be obtained by an Inverse Kinematics (IK) solver. The quality of the gripping pose is then evaluated by the performance of its trained policy using deep reinforcement learning as described in Section 4.3.

5.3.1 Inverse kinematics. We implement an optimization-based IK solver. The objective function is formulated for a position vector \mathbf{x} as follows:

$$\min_{\mathbf{q}} \sum_{i=1}^N \|f_i(\mathbf{q}) - \mathbf{p}_i(\mathbf{x})\|_2^2 + \text{clog}(\delta_i, 0.001) \quad (8)$$

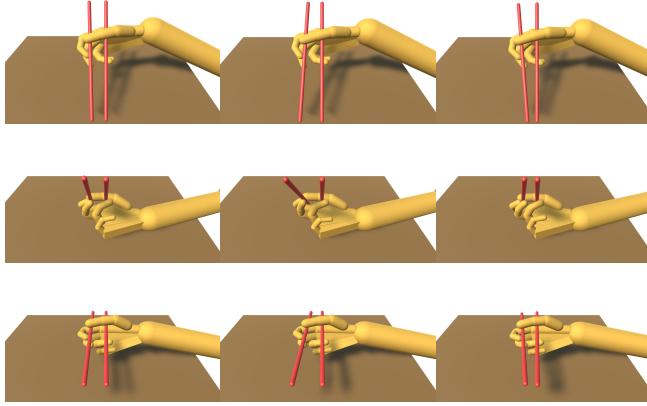


Fig. 5. For gripping pose evaluation, three one-second long motions are used to train the hand controller to open and close chopsticks while pointing to different directions.

where $\mathbf{p}_i(\mathbf{x})$ represents the 3D position of the contact point on the chopstick that finger i should touch, $f_i(\mathbf{q})$ calculates the 3D position of the point on fingertip i that is closest to $\mathbf{p}_i(\mathbf{x})$, and δ_i represents the depth of penetration between fingertip i and its contacting chopstick. $\text{clog}(\cdot)$ is a modified clamped log-barrier function defined as:

$$\text{clog}(z, z_0) = \begin{cases} -\frac{(z-z_0)^2}{z} \ln\left(\frac{z}{z_0}\right), & 0 < z < z_0, \\ 0, & z \geq z_0 \end{cases} \quad (9)$$

The first term in Equation 8 encourages fingers to touch chopsticks at the positions proposed by BO, and the second term penalizes potential penetrations between the fingers and the chopsticks. Note that we only explicitly control contacts between the chopsticks and the fingertips. Chopsticks contacts on other parts of the fingers or the hand, such as contacts with the valley between the index finger and the thumb, emerge naturally during our DRL policy learning.

We solve this IK problem using L-BFGS. As the objective function in Equation 8 is highly non-convex, we use the default pose shown in Figure 4 to initialize the IK solver. This effectively eliminates poor local minima, such as fingers touching the wrong side of the chopsticks.

5.4 Gripping Pose Evaluation

We evaluate each candidate gripping pose proposed by BO via deep reinforcement learning of a tracking control policy to accomplish basic chopsticks maneuvers without manipulating any object. The details of the DRL training can be found in Section 4. Since no objects are involved in these maneuvers, the corresponding reward term of Equation 6 is excluded from the reward function. Three one-second long motions as shown in Figure 5 are used for training, where the chopsticks open and close several times while pointing to different directions. We train each tracking policy for 500 epochs, then run the learned controller to perform all the test motions again. The average reward of the simulated motions, i.e., the undiscounted cumulative reward divided by the episode length, is sent back to BO as the quality score of the input candidate pose. More complicated

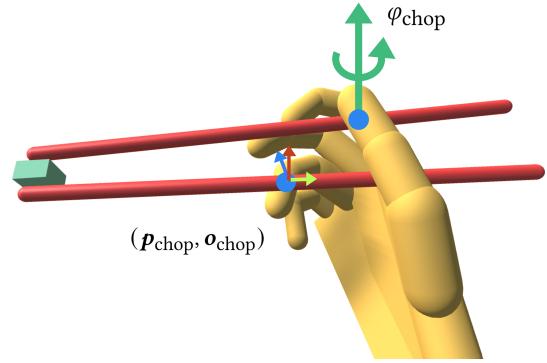


Fig. 6. The 7-DoF chopsticks model used in motion planning. The hand controllers still use the 12-DoF chopsticks model in simulation.

maneuvers could be used here for gripping pose evaluation, but the cost of computation will go up as well.

6 HIGH-LEVEL MOTION PLANNING

Given a relocation task characterized by an object to be moved and its target location, we propose a hierarchical control framework where a high-level motion planner is responsible for generating feasible kinematic motion trajectories for the hand, the chopsticks, and the object to accomplish the task. A low-level hand tracking controller, as described in Section 4, then tries to follow these trajectories to drive the simulated hand to move towards the object, pick it up and then drop it at the target location using the chopsticks.

The motion planner is queried once for each object to be relocated. It synthesizes feasible trajectories in two steps. First, it proposes a chopsticks configuration from a grasping model to ensure quality of the grasp, as will be detailed shortly in Section 6.1. The grasping model consists of pretrained neural network based models independent of the gripping styles and hand morphologies. Then a trajectory generation module computes the actual trajectories for the hand and arm, through trajectory optimization and inverse kinematics as will be described in Section 6.2. The trajectory generation algorithm takes as input the chopsticks configuration proposed by the grasping model, the object start and goal locations, as well as the hand morphology and desired gripping style.

The trajectories generated by the high-level motion planner are then passed to the low-level hand control policy for tracking. We train one hand controller for each chopstick gripping pose in the desired style by tracking a large set of trajectories created by the motion planner for moving objects between random start and goal locations. Once trained, the policy is robust enough to generalize to new trajectories generated by the same motion planner for new tasks not present in the training set.

6.1 Grasping Model

Determining how and where to grasp an object using chopsticks is a core mission of the motion planner, which is particularly challenging when dealing with objects of various shapes. We therefore simplify the planning problem in two ways. First, we reduce the

DoFs of the chopsticks from twelve to seven, similar to a parallel gripper as illustrated in Figure 6. This is based on the observation that humans usually hold the lower chopstick firmly and rotate the upper chopstick around the object to prepare for grasping. The configuration of the 7-DoF chopsticks can thus be described by a tuple $\mathbf{q}_{\text{chop}} = (\mathbf{p}_{\text{chop}}, \mathbf{o}_{\text{chop}}, \varphi_{\text{chop}})$, where \mathbf{p}_{chop} and \mathbf{o}_{chop} represent the position and orientation of the lower chopstick, and the scalar φ_{chop} represents the relative rotation of the upper chopstick with respect to the lower stick around the axis perpendicular to both sticks. Note that this simplified chopsticks model is only used in motion planning, and the full 12-DoF chopsticks are still used in simulation.

We further assume that the line connecting the two tips of the chopsticks go through the CoM of the object when grasped, which further reduces the planning problem to three DoFs, i.e., the orientation of the lower chopstick \mathbf{o}_{chop} . Once \mathbf{o}_{chop} is known, we can compute \mathbf{p}_{chop} from the object’s position and φ_{chop} from the object’s shape information.

We develop a neural-network based grasping model to predict the optimal grasping configuration for the chopsticks, from the shape and configuration of the object to be grasped. Our grasping model supports efficient planning and replanning at runtime, and is partially inspired by the GraspNet model proposed in [Fang et al. 2020a]. More specifically, our grasping model consists of two neural networks. Given the shape parameter of an input object, the *configuration network* nominates a number of candidate chopsticks configurations for grasping, together with their probabilities of success. The chopsticks configurations are specified in the local coordinate frame of the object in the configuration network. We then transform them into the global coordinate frame and pass them into a *reachability network* to estimate how likely each candidate configuration is within the reachable space of the simulated hand and arm. We also compute a continuity score for a configuration by measuring how close it is to the current chopsticks configuration in simulation. Closer chopstick configurations help produce natural hand and arm motions in sequential relocation tasks. We then multiply the probability from the configuration network, the score from the reachability network, and the continuity score all together as the final quality score of a candidate grasping configuration. The motion planner then selects the candidate configuration with the highest score for further trajectory planning.

6.1.1 Configuration network. There are usually multiple ways to grasp an object, especially for symmetric objects such as a ball. We thus design the configuration network to evaluate multiple candidate configurations simultaneously. Specifically, we uniformly discretize the three dimensional chopsticks configuration space into a set of N_c candidate configurations denoted as C_{chop} . As the configuration space corresponds to the orientation of the lower chopstick, this discretization can be performed easily using Euler angles. Then the configuration network takes the shape parameters of an object as input, and computes an N_c -dimensional vector representing the success probability for each configuration.

The configuration network is implemented as a fully-connected neural network with two 512-unit hidden layers and tanh as the activation function. A softmax layer is appended after the last linear

Table 1. The range of size of our tested geometry primitives.

	Width/Radius	Length	Height
Sphere	[0.5cm,1cm]		
Capsule	[0.5cm,1cm]	[2cm,4cm]	
Box	[1cm,2cm]	[1cm,2cm]	[1cm,2cm]

layer to turn its output into probabilities. The network is trained as a multi-class classification problem, matching a given object to configurations in C_{chop} with estimated success probabilities. The top n_c configurations will be passed to the reachability network for further assessment, as will be described shortly. In our implementation, we choose $N_c = 2000$ and $n_c = 10$.

We use synthetic data for our supervised learning problem. We first generate 100 objects from three primitive shapes and uniformly sample their sizes in the range as shown in Table 1. For each object, we then optimize for the grasping configuration using the particle swarm optimization algorithm with n_c random initial solutions [Miranda 2018]. The optimization objective follows the grasp quality metric employed by [Zhao et al. 2013], which encourages the center of the line connecting the chopsticks tips to stay close to the object CoM, and the direction of the connecting line to align with contact surface normals. Each of the optimized configurations is then mapped to its nearest neighbour in C_{chop} , for which the success probability in its corresponding one-hot vector is labeled as one.

6.1.2 Reachability network. The reachability network evaluates each candidate chopsticks configuration proposed by the configuration network, in terms of reachability in the global frame. We implement the reachability network as a fully-connected neural network with two 256-unit hidden layers, tanh activation functions, and a sigmoid output layer. Different from the configuration network, the reachability network is trained as a binary classification problem, which outputs the probability that whether a configuration is reachable by the hand and arm.

We use synthetic data for training as well. We first randomly sample 10000 chopsticks configurations in an operating cuboid of $0.5m \times 0.5m \times 0.25m$ over the table on which objects are placed. We then solve their corresponding arm poses using the analytical arm IK algorithm proposed in [Tolani and Badler 1996]. More specifically, from the chopsticks configuration and the chosen gripping pose and hand morphology, we can compute the desired rigid transformation for the hand, from which the IK algorithm then solves for a pose for the 7-DoF arm. If the hand transformation is reachable, the IK solver returns a solution, otherwise the IK solver returns no solution. For training, chopsticks configurations with an IK solution are labeled with probability one while configurations with no IK solutions are labeled with zero.

The reachability network could simply be replaced by the arm IK solver, which is relatively fast and generates solutions of acceptable quality. We choose the neural network based model for reachability tests mainly for the potential of switching to a more expensive IK algorithm in the future that not only tests the reachability of the desired end-effector transformations, but also assesses the naturalness of the solved arm poses. We note that in robotics, machine learning

models such as Gaussian mixture models are also used to rapidly predict a feasible catching configuration for object catching tasks [Kim et al. 2014].

6.1.3 Configuration Continuity. For a sequence of object relocation tasks, the hand and arm need to manipulate the chopsticks from one configuration to the next. The closer the chopsticks configurations are, the smoother the hand and arm trajectories will be, which improves the overall continuity and naturalness of the animation for consecutive relocation tasks. We thus calculate a continuity score between the candidate chopstick orientation and the current simulated chopstick orientation $\xi = \exp(-5 \cdot \Theta(\mathbf{o}_{\text{chop}}, \mathbf{o}_{\text{chop}}^0))$, where $\mathbf{o}_{\text{chop}}^0$ is the simulated lower chopstick orientation when the motion planner is activated for a new object relocation task. We then multiply the continuity score with the grasping success probability and the reachability score as the final quality score estimated for a candidate configuration.

6.2 Trajectory Generation

Once the optimal chopsticks configuration is selected by the grasping model, the motion planner needs to generate trajectories for the chopsticks, the hand and arm, and the object. This is done in three steps. First, a collision-free trajectory for the chopsticks is computed through trajectory optimization. Then the arm trajectory is solved by the arm IK algorithm, which also considers continuity when applied to a sequence of IK problems [Tolani and Badler 1996]. Lastly, the object trajectory is easily synthesized from the tips of the chopsticks. As the latter two steps are straightforward, we will only discuss our trajectory optimization method for the chopsticks.

Following the conventions in hand manipulation research, we segment an object relocation task into three phases: *approaching* the object, *relocating* the object, and then *releasing* the object. In the *releasing* phase, we simply keep the transformation of the chopsticks fixed and set φ_{chop} to 0. We utilize trajectory optimization for the first two phases. Two additional control points are inserted between the start and end chopsticks configurations, then the chopsticks positions are computed by a degree-three Béizer curve from the four control points, and the chopsticks orientations are computed with spherical linear interpolation between two control points. We then solve for the optimal control points that generate the shortest collision-free trajectory, which can be formulated as

$$\min_{\mathbf{q}_1, \mathbf{q}_2} \int_t \|\dot{\mathbf{m}}(t)\| dt + \text{clog}(\delta(\mathbf{m}(t)), 0.001) dt \quad (10)$$

where $\mathbf{m}(t) = \mathbf{m}(t; \mathbf{q}_1, \mathbf{q}_2)$ represent the trajectory defined by the control points $\mathbf{q}_1, \mathbf{q}_2$, and δ computes the penetration depth between the chopsticks and the environment in trajectory $\mathbf{m}(t)$, and $\text{clog}(\cdot)$ is the same barrier function as defined in Equation 9. The duration of this trajectory is computed as $d_{\text{chop}}/\tilde{v}_{\text{chop}}$, where d_{chop} is the length of the displacement vector of the chopsticks in the current phase, and \tilde{v}_{chop} is a hyperparameter indicating the desired speed of the chopsticks. We use $\tilde{v}_{\text{chop}} = 0.25 \text{ m/s}$ for our demo results. We use numerical integration with a discretized time step 10ms to compute the integral in Equation 10. We use the L-BFGS algorithm with multi-start points to solve the optimization, similar to [Li et al. 2011].

Fig. 7. Our simulated hand has 30 DoFs in total. The thumb has 6 DoFs, and the other fingers each has 4 DoFs. The four bones connecting the root of the hand to the base of the fingers, indicated as green arrows, each has 2 DoFs to model small deformations of the palm.

7 RESULTS

We implement our system with Pytorch version 1.9.0 [Paszke et al. 2017] and the MuJoCo physics simulator version 1.5 [Todorov et al. 2012]. We use two 6-DoF capsules to model the chopsticks. The length and radius of the chopsticks are 26 cm and 0.4 cm respectively. Our hand model as shown in Figure 7 has 30 DoFs in total, and shares the same joint hierarchy as the hand in [Ye and Liu 2012]. The only difference is that the palm part connecting the thumb and index finger, the so-called union valley in acupuncture, is modeled with a capsule instead of a box. We found that the chopsticks need stable contacts with the valley to perform well, which aligns with human experiences. We mount the hand on a 7-DoF robot arm consisting of a shoulder, an elbow, and a wrist joint. We run the simulation and control at the same frequency at 100 Hz. Torque limits for hand joints are taken from values of the shadow hand robot used in [Rajeswaran et al. 2018].

For low-level DRL policy training, we set λ to 0.95 for both TD(λ) and GAE(λ). The discount factor γ is set to 0.99 and the PPO policy clip ratio is 0.2. The learning rates of the actor and critic network are set to 3×10^{-5} and 3×10^{-4} , respectively. We sample 1×10^4 state-action tuples with 32 parallel simulation environments for each training epoch, as all experiments are performed on a workstation with 32 Intel-i9-9980XE CPUs. Both networks are updated 10 times in each epoch with a mini-batch size of 256. During training the action noise linearly decreases from 0.1 to 0.01 in the first 5×10^7 simulation steps to encourage exploration in early training and exploitation in later training. For learning basic chopsticks maneuvers of fixed length in the stage of gripping pose optimization, we run PPO for 500 epochs. For learning object relocation, we gradually increase the episode length to facilitate policy learning similar to [Peng et al. 2018a]. More specifically, the episode length is 100 control steps for the first 5000 epochs, and then increases by 100 steps every 500 epochs. The tracking trajectories are generated by the motion planner for 100 multi-object relocation tasks and last for about 30 minutes in duration. Each multi-object relocation task contain eight objects to be moved between random locations. We run PPO for 2×10^4 epochs for each gripping pose in a particular style, which takes roughly two days on our workstation.

For high-level motion planning, the planner is queried whenever a new object is required to be moved. The planning takes roughly $3 \sim 5$ seconds using a non-optimized single-thread implementation on our workstation. The computation bottleneck is the trajectory optimization component. It is possible to further improve the efficiency of the optimizer, but real-time performance may be hard to achieve using the current algorithms.

We show optimized chopsticks gripping poses in different styles in Section 7.1. Various learned chopsticks skills are demonstrated with multiple hand morphologies in Sections 7.2, 7.3, and 7.5. We demonstrate the ability to learn to use other tools with the same

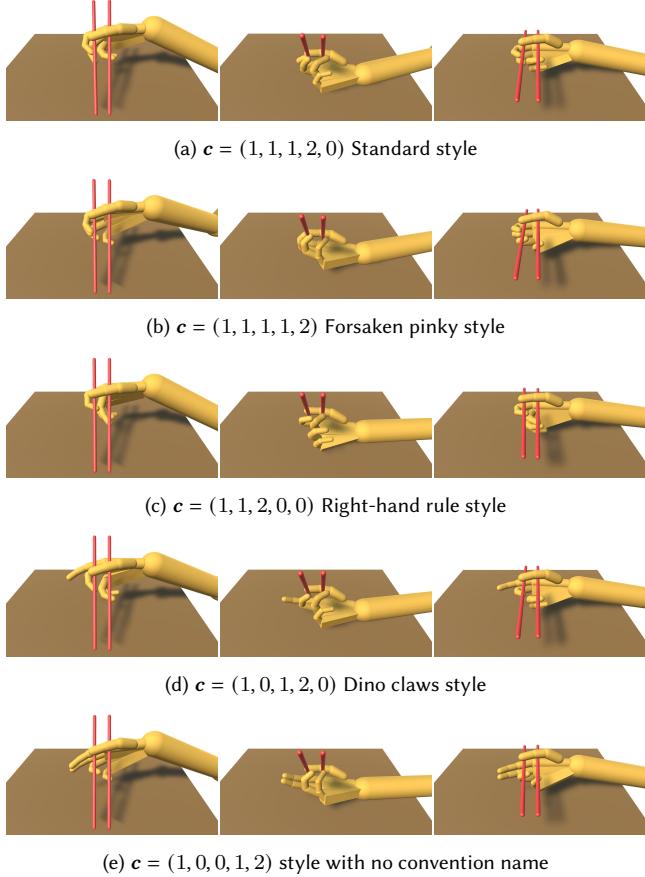


Fig. 8. Visualization of the optimized chopsticks gripping poses in five styles, performing the basic open-and-close chopsticks maneuver.

framework in Section 7.4. Lastly in Section 7.6 we conduct ablation studies and comparisons to validate each component of our framework. We encourage readers to watch our supplementary videos to better comprehend the quality of our learned chopsticks skills.

7.1 Diverse Chopsticks Gripping Styles

The Bayesian Optimization takes roughly six days in total to optimize gripping poses for the seventeen valid gripping styles. In Figure 8 we show the most distinctive five styles: $(1, 1, 1, 2, 0)$, $(1, 1, 1, 1, 2)$, $(1, 1, 2, 0, 0)$, $(1, 0, 1, 2, 0)$ and $(1, 0, 0, 1, 2)$. Four out of the five styles are documented in [Macro 2021], following which we name the gripping styles. The standard style with $c = (1, 1, 1, 2, 0)$ is the most common way to use chopsticks, and also considered the best way by common belief. Our BO results verify that it is indeed the most efficient way to use chopsticks, as policy learning for the standard style converges the fastest with the highest normalized return, as shown in Figure 17. The optimized gripping poses for the remaining twelve styles are shown in Figure 22 in Appendix B.

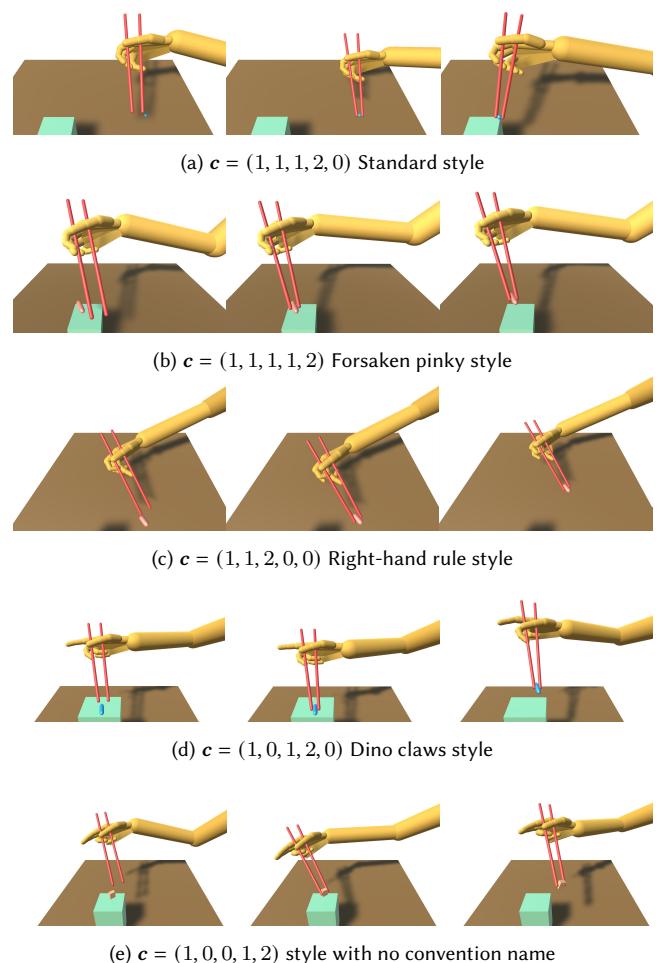


Fig. 9. The hand controls chopsticks to grasp and move various objects with different gripping poses.

7.2 Chopsticks Skills for Object Relocation

We evaluate our framework with object relocation tasks where the simulated hand uses the chopsticks to grasp objects of various shapes and sizes, and then move or throw them to some desired target locations.

7.2.1 Grasp and Move. We train hand control policies using sequential grasp-and-move object relocation tasks. Figure 9 shows the learned chopsticks skills using different gripping styles. We also test the generalization ability of the learned controllers with a more challenging stacking task, as shown in the teaser Figure 1a. This task is not included during policy training, yet the learned controllers can directly finish the stacking task without any fine tuning.

7.2.2 Grasp and Throw. Throwing is an efficient means to relocate objects when the target position is out of reach by the arm and hand. Our motion planner generates kinematic chopsticks trajectories for throwing tasks following the method described in [Zeng et al. 2020].

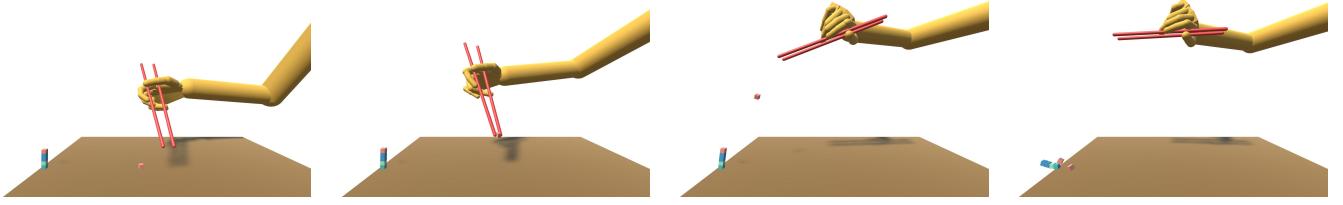


Fig. 10. Grasping and throwing a box to hit another stack of boxes.

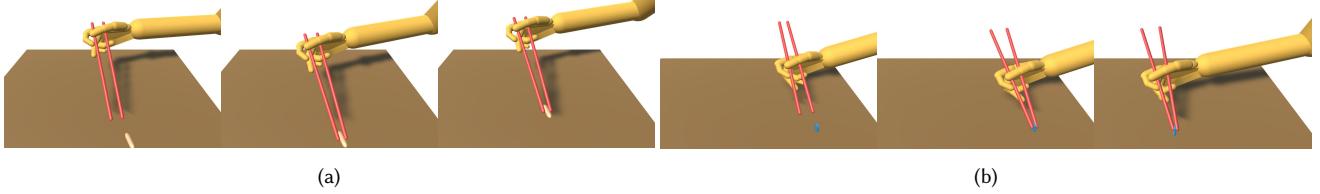


Fig. 11. Our controllers are parameterized so that the hand can hold the chopsticks high or low.

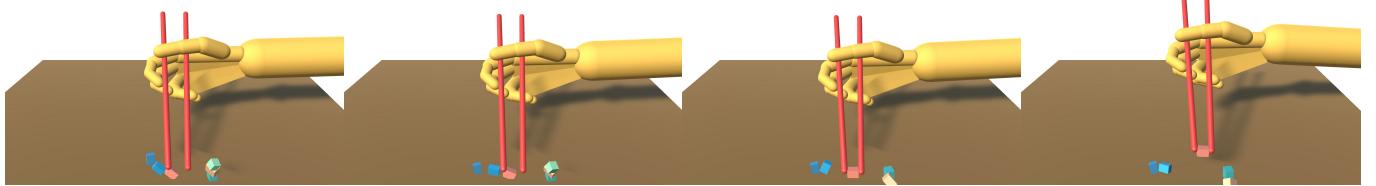


Fig. 12. Grasping a moving object without replanning.

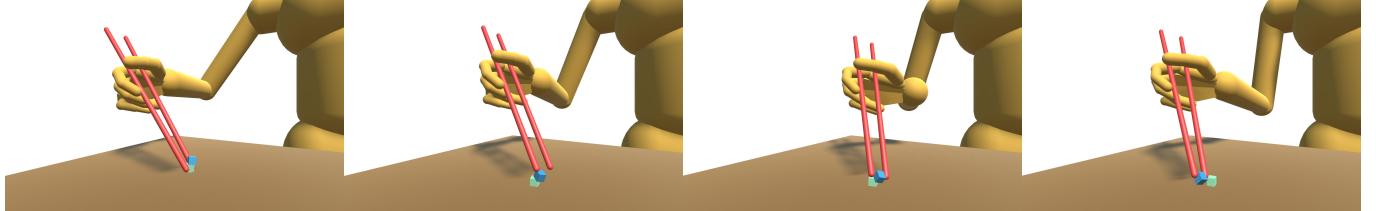


Fig. 13. Relocating two boxes together. The top blue box falls down at the end of the first move. With replanning, the chopsticks are able to grasp the top box during falling to continue the second move.

More specifically, we first estimate the needed position and velocity of the chopsticks at the end of the relocating phase, given the final target position of the object. Then we use trajectory optimization for the relocating phase to first move the chopsticks to the estimated releasing position and then rotate them to achieve the estimated releasing velocity. Trajectory generation for the approaching and releasing phases are the same as described in Section 6.2. The trajectory of the object after being thrown is assumed to be projectile, and aerodynamic drags are ignored. We train the hand control policy by tracking the above planned trajectories, the same as for the grasp-and-move task. Figure 10 shows one of our throwing results where a box is grasped and thrown to hit another stack of boxes far away.

7.2.3 Parameterized Controllers for Different Holding Positions. Humans can hold chopsticks at different positions in similar poses. For example, children and beginners tend to hold chopsticks near the tips for easier control. Expert users hold chopsticks near the rear ends to increase the reachability and avoid potential harmful injury from hot food. Our system can learn such parameterized controllers with ease.

We parameterize the holding position of the chopsticks by translating the chopsticks along the axial direction of the lower chopstick. Such translations do not change the relative rotation between the chopsticks and the palm. The holding position is also added into the input state representation of the hand control policy to enable learning of parameterized controllers. Figure 11 shows that the learned controllers can hold the chopsticks at multiple positions to relocate

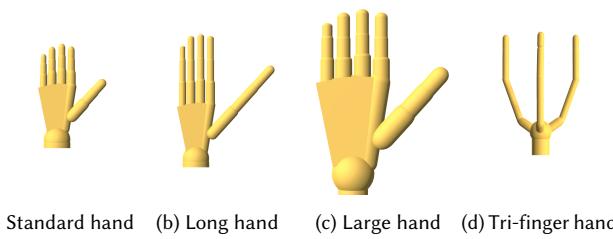


Fig. 14. Our framework works well for drastically different hand morphologies.

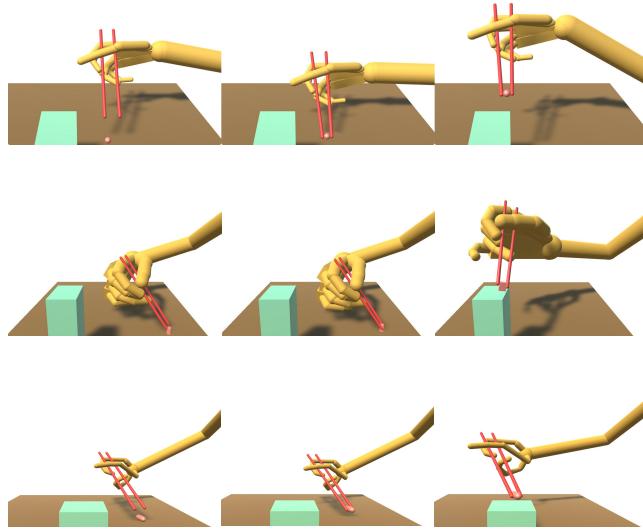


Fig. 15. Chopsticks skills learned for hands of different morphologies.

objects. We allow a range of ± 5 cm for the chopsticks translation in our tests. Note that policies using some holding positions are harder to train than others, which leads to biased sampling during DRL training. We adopt the adaptive sampling strategy similar to [Won and Lee 2019] to increase samples for harder tasks.

7.3 Diverse Hand Morphologies

Our framework can learn chopstick skills for drastically different hand morphologies. In Figure 14 we visualize three additional hand models that we have tested: a hand with long fingers, a large hand, and a tri-finger hand. The fingers of the long hand double the lengths of the fingers of the standard hand. The large hand doubles the size of the standard hand in all dimensions for both the fingers and the palm. The tri-finger hand is designed by ourselves following claw toy grabbers commonly seen in arcade machines. We show some of the learned skills for each hand in the teaser Figures 1c, 1d, and Figure 15. We are happy to find that the tri-finger hand can learn to use chopsticks successfully as well. Note that we still use the same arm model for these different hands, although we scale the radius of the arm to match the dimension of the hands for better

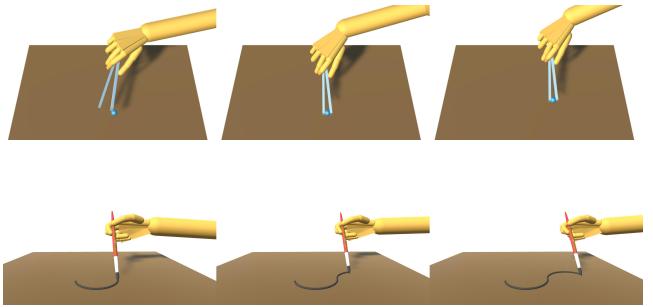


Fig. 16. Top: relocating a ball using a pair of tongs. Bottom: tracing a curve using a brush.

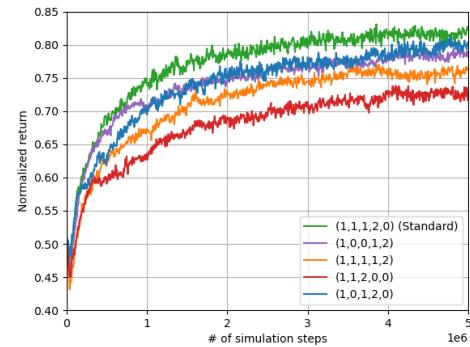


Fig. 17. Training curves of learning basic chopsticks maneuvers using different gripping styles. The standard style is indeed the most efficient way of using chopsticks.

visualization. Changing the length of the arm model mainly affects the reachability of the hand and does not affect the ability to use chopsticks by hand.

7.4 Using Other Tools

Our framework can be applied to learn physics-based hand controllers to use other types of tools. In Figure 16 we demonstrate relocating a ball with a pair of tongs, and tracing a curve using a brush. The tongs are modeled as a 7-DoF gripper, with its two bars modeled as $26\text{cm} \times 0.8\text{cm} \times 2\text{cm}$ cuboids. We model the pivot of the tongs with an angular spring to provide restoring torques. More specifically, the torques are computed as $-k(\theta - \theta_{\text{rest}})$, where $k = 20\text{Nm}$ is the stiffness parameter, and $\theta_{\text{rest}} = 0.2\text{rad}$ is the resting angle between the two bars. The brush is modeled as a long capsule of 26cm in length and 0.4cm in radius.

We follow the same pipeline designed to learn chopsticks skills to learn tongs skills. The only difference is that we only allow the thumb, index, and middle fingers to contact the tongs. For curve tracing with the brush, we use a much simplified version of the motion planner. As there is no tool-object interactions involved, the grasping model to predict a configuration for the tool to grasp the object is not needed. Trajectory generation for the tool is also much

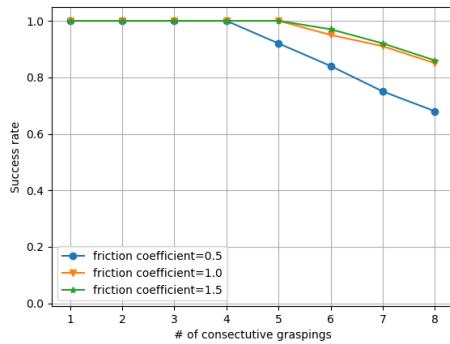


Fig. 18. The task success rate with respect to the number of consecutive object relocations performed. We test three object-chopsticks friction coefficients, around the default MuJoCo friction coefficient 1.0.

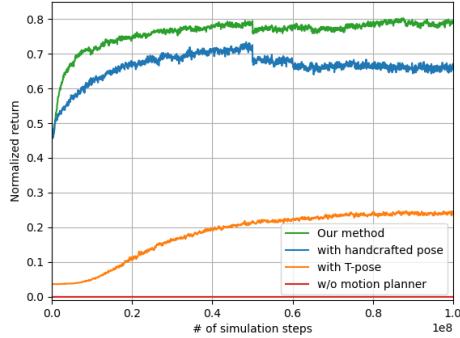


Fig. 19. Learning curves for policies trained with different ablations. Without the high-level motion planner, the policy does not learn at all. Using the default T-pose shown in Figure 4, the policy learns very poorly. Our system can also learn using a handcrafted gripping pose, but using the BO-optimized pose performs better. Note that the performance drop around the middle of the training is due to lengthening the training episodes gradually from then on.

easier, by simply tracing the curves using the tool tip, and the hand and arm trajectories can be solved directly from IK.

7.5 Robustness

Our learned physics-based hand controllers can handle noises, perturbations, and uncertainties to certain extent. Figure 12 shows such an example. The target object rotates due to unexpected collisions when the chopsticks approach the object. The controller can still grasp the rotated box successfully according to the plan computed before the collision. Figure 13 shows another example of relocating two stacked boxes together. The chopsticks only grasp the bottom box during the first move and the top box starts to fall when the first move ends. The controller can grasp the top box during falling according to the plan made at the beginning of the second move.

Generally speaking, the success rate of chopsticks-based grasps depends on many factors, such as the weight of the objects, the shape of the chopsticks, the hand gripping styles etc. Here we report the grasping success rates with respect to the object-chopsticks friction coefficients for consecutive relocation tasks. More specifically, we use our motion planner to generate 50 sequential grasp-and-move tasks, each containing eight random objects to be relocated. The friction coefficients are set to one of three values around the MuJoCo default 1.0. The success rates are plotted in Figure 18. Interestingly enough, we find our hand controllers always succeed for the first few grasps. Failures only occur at the latter stage. Objects with smaller friction coefficients are indeed more difficult to be grasped. Another contributing factor to the failures is likely to be the soft contact model implemented by MuJoCo as discussed in Section 7.6.3, as we observe failure cases where the chopsticks gradually slip out of the hand after a few moves.

To handle failure cases and bigger perturbations, replanning by the motion planner according to the latest scene configuration is needed. We note that human-level performance is not 100% for chopsticks-based grasping tasks either, especially for tiny or slippery objects, so replanning is also seen in real-life scenarios. We can simulate such behavior by deliberately adding Gaussian noises $\mathcal{N}(0, \sigma^2 I)$ to \mathbf{p}_{chop} of the chopsticks configurations computed by the grasping model. $\sigma = 3\text{mm}$ in our experiments. The hand controller tracks the noisy motion plans, which eventually leads to failures in sequential relocation tasks. Upon such failures, we replan again with no added noise. Then the object can be grasped successfully. We encourage the readers to refer to the “planning with noise” examples in the supplementary video.

7.6 Ablation and Comparison

We conduct ablation and comparative studies to justify and validate two major components of our control and learning framework: the hierarchical control structure, and the gripping pose optimization module. Additionally, we report the effect of related MuJoCo contact dynamics parameters.

7.6.1 Hierarchical Control. Our high-level motion planner computes kinematic motion trajectories for the low-level hand controllers to track, which greatly simplifies the DRL reward design and improves the overall motion quality without using pre-captured example data. For system ablation without using the motion planner, we use a sparse reward to directly measure the success of a task at the very end of the task. More specifically, we return a reward 1.0 if the object can reach the target position within 0.5cm in distance, and a reward 0.0 otherwise. We use exactly the same PPO algorithm with the same parameter and hyperparameter settings. As shown in Figure 19, this scheme does not learn at all, with close to zero returns after 100 million simulation steps. The learned hand controller just randomly moves the fingers and cannot even hold the chopsticks firmly, let alone using chopsticks to grasp objects.

7.6.2 Gripping Pose Optimization. We ablate the gripping pose optimization with two alternatives: policy learning with the default T-pose and a handcrafted gripping pose. The T-pose has no preferred finger-chopsticks contact relationships, so the reward term r_{contact} is

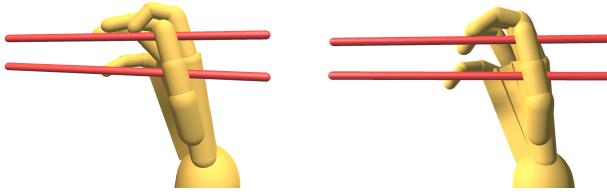
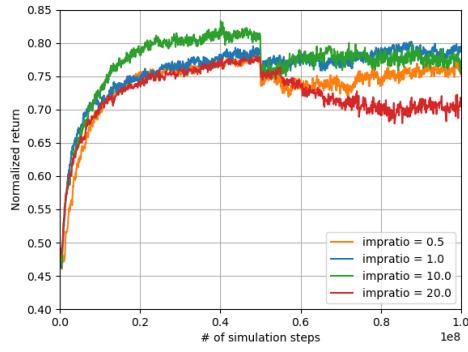
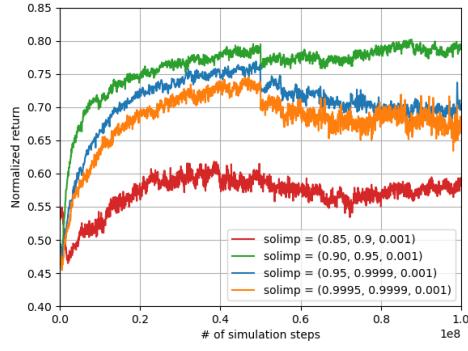


Fig. 20. A handcrafted pose (left) vs. our optimized gripping pose (right) for the standard gripping style.



(a) *solimp* controls the contact constraint impedance. The default value performs the best in terms of policy learning.



(b) *impratio* controls the frictional-to-normal constraint impedance. Mid-range values perform the best in terms of policy learning.

Fig. 21. Policy training curves for different values of contact dynamics parameters in MuJoCo.

not used in DRL training. The handcrafted gripping pose, as shown in Figure 20 left, is tuned to follow reference pictures in [Macro 2021]. As shown in Figure 19, policy training with T-pose progresses poorly. The final policy cannot even hold the chopsticks steadily. Policy learning with the handcrafted pose shows better performance, but still converges slower and has inferior final policy, compared to learning with our BO optimized pose as shown in Figure 20 right.

7.6.3 Contact Dynamics. Chopsticks skills are contact-rich control problems for which contact dynamics plays an important role in the success, robustness, and quality of the learned controllers. We train our hand controllers and synthesize the demo animations using the MuJoCo default simulation parameters. However, the synthesized motions contain visible artifacts including penetrations of the chopsticks into the fingers, and objects being moved and stacked looking too soft or sticky. We therefore explore more settings for a few contact dynamics parameters in MuJoCo.

Contacts in MuJoCo are inherently soft and modeled as a convex optimization rather than the conventional LCP (Linear Complementarity Problem) [Todorov 2014]. The default parameters are also set to prefer soft contacts to encourage stable simulations. In particular, the parameter *solimp* parameterizes the impedance, which controls how “hard” the contact constraints are. The default setting of *solimp* is (0.90, 0.95, 0.001). The valid range of the first two values of *solimp* is [0.0001, 0.9999]. The larger they are, the harder the contact constraints will be. In Figure 21a we show the policy learning curves for different settings of *solimp*. Policy training with the default setting does perform the best, although at the cost of visible penetrations. However, it does not mean that the more penetrations allowed, the better the policy will learn. Policy learning with (0.85, 0.9, 0.001) actually fails due to large penetrations between the fingers and chopsticks.

A related issue caused by MuJoCo’s soft contact model is that gradual contact slip cannot be avoided, even with large friction coefficients. There is a parameter *impratio* that determines the ratio of frictional-to-normal constraint impedance for friction cones. Settings larger than 1 cause friction forces to be “harder” than normal forces, having the general effect of reducing slip. We train the low-level control policy for the standard gripping style with different *impratio* values centered around its default value 1.0. As shown in Figure 21b, policy learning with mid-range values have similar final performance, while policy learning with too small or too large values shows worse performance.

8 CONCLUSION AND DISCUSSION

We have presented a physics-based learning and control system for object manipulation using chopsticks. This is a challenging task that involves complex interactions between the hand, chopsticks, and objects. Our key insight is to tackle the problem by solving two sub-problems: finding good grips to hold the chopsticks stably first, and then using them to grasp and relocate objects. More specifically, the gripping pose is a strong determinant of successful chopstick use later, and we use Bayesian optimization to efficiently explore the gripping pose space. Candidate gripping styles are evaluated through deep reinforcement learning on basic chopsticks maneuvers. After good gripping poses are found through BO and DRL, the actual object relocation using chopsticks is learned in two stages. A high-level motion planner first generates kinematic trajectories for the chopsticks and hand to satisfy task requirements. Then low-level hand control policies are trained to track the generated motion plans using a chosen gripping pose through DRL again. Whenever possible, we adopt common design choices, such as PPO for DRL learning, and MuJoCo with default parameters for simulation.

We have demonstrated physics-based object relocation using chopsticks in diverse gripping styles for multiple hand morphologies, for the first time in the literature. Our framework does not need any motion capture data or human demonstrations, and is easily applicable to virtual creatures and robotic manipulators. The framework is also transferable to learning other tools such as tongs, tweezers, pens and brushes, with minor tool-specific changes for style pruning and trajectory generation. Manipulating scissors without actually cutting things is also doable, although real cutting with scissors requires additional components such as a thin-shell simulator and a deformable object motion planner.

We have focused on learning visually robust policies with a limited number of quantitative tests. We hope our paper will stimulate future work on more systematic investigations on factors and variations that can affect the success rates of chopsticks-based object manipulations tasks, such as object shapes and sizes, object weights, material and friction properties of the chopsticks and objects, chopsticks geometries, and gripping styles. Different robustness metrics other than the task success rates are also worth exploring.

There are many limitations of our current solution that we would like to further investigate in the near future. We use a 7-DoF parallel gripper to simplify the kinematic planning of chopsticks movements. However, some gripping styles allow chopsticks to move more freely, such as the “dangling stick” and the “Italian” style shown in Figure 2. To reproduce such styles, we need to model the chopsticks as a more versatile gripper with higher DoFs in the motion planner. Another possible solution is to learn a low-dimensional chopsticks motion manifold, similar to the approach described in [Holden et al. 2015] for learning a full-body human motion manifold, and train the grasping model in the latent space. Such a data-driven approach may produce more natural and diverse chopsticks manipulations, at the cost of pre-capturing all the chopsticks skills beforehand.

Our learned controllers cannot use chopsticks to successfully manipulate objects for an infinitely long time. The chopsticks gradually slip in hand after relocating objects repeatedly, and this destroys the good grips. Most likely this is caused by the MuJoCo contact dynamics models. As far as we know, no rigid body contact dynamics solver can completely avoid unstable contacts or contact slips. Modelling fingers and palms as truly soft bodies as in [Jain and Liu 2011] may lead to more robust skills. In the long run, we would like to learn to adjust chopsticks grips with dexterous finger gating maneuvers, such as picking up chopsticks from a table top, changing grips, and moving chopsticks up and down in hand.

Currently our system runs at interactive rates but not realtime, bottlenecked at the motion planner. An idea for improvement is to replace the trajectory optimization component with a neural network model, which may be trained jointly with the low-level control policies, similar to the ideas of [Peng et al. 2017] and [Zeng et al. 2019]. We would also like to explore the possibility of combining gripping pose optimization and control policy learning into one iterative learning pipeline, which alternates the training between optimizing gripping poses and improving low-level control policies, evaluated on the same set of object relocation tasks. Such a scheme may potentially achieve better learning efficiency and control performance.

We have only touched the tip of the iceberg for physics-based chopsticks skills. How to eat and cut noodles with chopsticks? How to flip a piece of meat using chopsticks? How to beat an egg with chopsticks? Is it easier to use Chinese, Japanese, or Korean chopsticks for a certain type of food? Modeling and planning with soft or amorphous materials as in [Zhang et al. 2021] is a must. Last but not least, we would like to transfer our chopsticks controllers to real anthropomorphic robot hands using “Sim2Real” techniques such as domain randomization and adaptation similar to [Peng et al. 2018b, 2020].

ACKNOWLEDGMENTS

We would like to thank the anonymous reviewers for their constructive suggestions and feedback. We also thank Zhiqi Yin, Yujie Wang, and Michiel van de Panne for various discussions and help. Yin is partially supported by NSERC Discovery Grants Program RGPIN-06797 and RGPAS-522723.

REFERENCES

- Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, et al. 2019. Solving rubik’s cube with a robot hand. *arXiv preprint arXiv:1910.07113* (2019).
- Sheldon Andrews and Paul G Kry. 2013. Goal directed multi-finger manipulation: Control policies and analysis. *Computers & Graphics* 37, 7 (2013), 830–839.
- Yahya Aydin and Masayuki Nakajima. 1999. Database guided computer animation of human grasping using forward and inverse kinematics. *Computers & Graphics* 23, 1 (1999), 145–154.
- Kevin Bergamin, Simon Clavet, Daniel Holden, and James Richard Forbes. 2019. DReCon: Data-driven responsive control of physics-based characters. *ACM Transactions on Graphics (TOG)* 38, 6, Article 206 (2019).
- Eric Brochu, Tyson Brochu, and Nando de Freitas. 2010. A Bayesian interactive optimization approach to procedural animation design. In *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. 103–112.
- Eric Brochu, Abhijeet Ghosh, and Nando de Freitas. 2007. Preference galleries for material design. *SIGGRAPH Posters* 105, 10.1145 (2007), 1280720–1280834.
- Bao-Chi Chang, Biing-Shiu Huang, Ching-Kong Chen, and Shyh-Jen Wang. 2007. The pincer chopsticks: The investigation of a new utensil in pinching function. *Applied ergonomics* 38, 3 (2007), 385–390.
- Tao Chen, Jie Xu, and Pukit Agrawal. 2021. A system for general in-hand object re-orientation. In *Conference on Robot Learning*. PMLR.
- George ElKoura and Karan Singh. 2003. Handrix: animating the human hand. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. 110–119.
- Hao-Shu Fang, Chenxi Wang, Minghao Gou, and Cewu Lu. 2020a. Graspnet-1billion: A large-scale benchmark for general object grasping. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)*. 11444–11453.
- Kuan Fang, Yuke Zhu, Animesh Garg, Andrey Kurenkov, Viraj Mehta, Li Fei-Fei, and Silvio Savarese. 2020b. Learning task-oriented grasping for tool manipulation from simulated self-supervision. *The International Journal of Robotics Research* 39, 2-3 (2020), 202–216.
- Guillermo Garcia-Hernando, Edward Johns, and Tae-Kyun Kim. 2020. Physics-based dexterous manipulations with estimated hand poses and residual reinforcement learning. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 9561–9568.
- GPy. since 2012. GPy: A Gaussian process framework in python. <http://github.com/SheffieldML/GPy>.
- Philipp Hennig and Christian J Schuler. 2012. Entropy search for information-efficient global optimization. *The Journal of Machine Learning Research* 13, 1 (2012), 1809–1837.
- Daniel Holden, Jun Saito, Taku Komura, and Thomas Joyce. 2015. Learning motion manifolds with convolutional autoencoders. In *SIGGRAPH Asia 2015 Technical Briefs*. 1–4.
- Sha Hu, Zeshi Yang, and Greg Mori. 2021. Neural fidelity warping for efficient robot morphology design. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 7079–7086.
- Zhiyong Huang, Ronan Boulic, Nadia Magnenat Thalmann, and Daniel Thalmann. 1995. A multi-sensor approach for grasping and 3D interaction. In *Computer graphics*. Elsevier, 235–253.

- Sumit Jain and C Karen Liu. 2011. Controlling physics-based characters using soft contacts. *ACM Transactions on Graphics (TOG)* 30, 6 (2011), 1–10.
- Rohan A Joseph, Alvin C Goh, Sebastian P Cuevas, Michael A Donovan, Matthew G Kauffman, Nilson A Salas, Brian Miles, Barbara L Bass, and Brian J Dunkin. 2010. “Chopstick” surgery: A novel technique improves surgeon performance and eliminates arm collision in robotic single-incision laparoscopic surgery. *Surgical endoscopy* 24, 6 (2010), 1331–1335.
- Korrawe Karunratnakul, Jinlong Yang, Yan Zhang, Michael J Black, Krikamol Muandet, and Siyu Tang. 2020. Grasping field: Learning implicit representations for human grasps. In *2020 International Conference on 3D Vision (3DV)*. IEEE, 333–344.
- Liyiming Ke, Ajinkya Kamat, Jingqiang Wang, Tapomayukh Bhattacharjee, Christoforos Mavrogiannis, and Siddhartha S Srinivasa. 2020. Telemanipulation with chopsticks: Analyzing human factors in user demonstrations. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 11539–11546.
- Liyiming Ke, Jingqiang Wang, Tapomayukh Bhattacharjee, Byron Boots, and Siddhartha Srinivasa. 2021. Grasping with chopsticks: Combating covariate shift in model-free imitation learning for fine manipulation. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 6185–6191.
- Junhwan Kim, Frederic Cordier, and Nadia Magnenat-Thalmann. 2000. Neural network-based violinist’s hand animation. In *Proceedings Computer Graphics International 2000*. IEEE, 37–41.
- Seungsu Kim, Ashwini Shukla, and Aude Billard. 2014. Catching objects in flight. *IEEE Transactions on Robotics* 30, 5 (2014), 1049–1065.
- Uikyum Kim, Dawoon Jung, Heeyeon Jeong, Jongwoo Park, Hyun-Mok Jung, Joono Cheong, Hyouk Ryeol Choi, Hyunmin Do, and Chanhun Park. 2021. Integrated linkage-driven dexterous anthropomorphic robotic hand. *Nature Communications* 12, 1 (2021), 1–13.
- Yoshihito Koga, Koichi Kondo, James Kuffner, and Jean-Claude Latombe. 1994. Planning motions with intentions. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*. 395–408.
- Yuki Koyama, Issei Sato, and Masataka Goto. 2020. Sequential gallery for interactive visual design optimization. *ACM Transactions on Graphics (TOG)* 39, 4, Article 88 (2020).
- Paul G Kry and Dinesh K Pai. 2006. Interaction capture and synthesis. *ACM Transactions on Graphics (TOG)* 25, 3 (2006), 872–880.
- Shutao Li, Mingkui Tan, Ivor W Tsang, and James Tin-Yau Kwok. 2011. A hybrid PSO-BFGS strategy for global optimization of multimodal functions. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 41, 4 (2011), 1003–1014.
- C Karen Liu. 2008. Synthesis of interactive hand manipulation. In *Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. 163–171.
- C Karen Liu. 2009. Dexterous manipulation from a grasping pose. *ACM Transactions on Graphics (TOG)* 28, 3, Article 59 (2009).
- Libin Liu and Jessica Hodgins. 2017. Learning to schedule control fragments for physics-based characters using deep Q-learning. *ACM Transactions on Graphics (TOG)* 36, 3, Article 42a (2017).
- Macro. since 2021. Ten thousand ways to use chopsticks. <https://marcosticks.org/poster-thousand-ways-to-use-chopsticks/>.
- Josh Merel, Saran Tunyasuvunakool, Arun Ahuja, Yuval Tassa, Leonard Hasenclever, Vu Pham, Tom Erez, Greg Wayne, and Nicolas Heess. 2020. Catch & Carry: Reusable neural controllers for vision-guided whole-body tasks. *ACM Transactions on Graphics (TOG)* 39, 4, Article 39 (2020).
- Lester James V. Miranda. 2018. PySwarms, a research-toolkit for particle swarm optimization in Python. *Journal of Open Source Software* 3 (2018).
- Igor Mordatch, Zoran Popović, and Emanuel Todorov. 2012. Contact-invariant optimization for hand manipulation. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. 137–144.
- Yukiko Mukai and Keiko Hashimoto. 1978. A study on ways of holding chopsticks. *Journal of Home Economics of Japan* 29, 7 (1978), 467–473.
- Anusha Nagabandi, Kurt Konolige, Sergey Levine, and Vikash Kumar. 2020. Deep dynamics models for learning dexterous manipulation. In *Conference on Robot Learning*. PMLR, 1101–1112.
- Tomoko Osera, Chihiro Yamamoto, Rika Senke, Misako Kobayashi, Setsuko Tsutie, and Nobutaka Kurihara. 2018. Relationship between mothers and children on how to hold chopsticks and concerns about chopsticks in Japanese kindergarten. *Journal of Japanese Society of Shokuiku* 12, 1 (2018), 19–25.
- Soohwan Park, Hoseok Ryu, Seyoung Lee, Sunmin Lee, and Jehee Lee. 2019. Learning predict-and-simulate policies from unorganized human motion data. *ACM Transactions on Graphics (TOG)* 38, 6, Article 205 (2019).
- Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in pytorch. (2017).
- Xue Bin Peng, Pieter Abbeel, Sergey Levine, and Michiel van de Panne. 2018a. Deepmimic: Example-guided deep reinforcement learning of physics-based character skills. *ACM Transactions on Graphics (TOG)* 37, 4, Article 143 (2018).
- Xue Bin Peng, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. 2018b. Sim-to-real transfer of robotic control with dynamics randomization. In *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE, 3803–3810.
- Xue Bin Peng, Glen Berseth, KangKang Yin, and Michiel Van De Panne. 2017. Deeploco: Dynamic locomotion skills using hierarchical deep reinforcement learning. *ACM Transactions on Graphics (TOG)* 36, 4, Article 41 (2017).
- Xue Bin Peng, Michael Chang, Grace Zhang, Pieter Abbeel, and Sergey Levine. 2019. MCP: Learning composable hierarchical control with multiplicative compositional policies. *Advances in Neural Information Processing Systems* 32 (2019).
- Xue Bin Peng, Erwin Coumans, Tingnan Zhang, Tsang-Wei Edward Lee, Jie Tan, and Sergey Levine. 2020. Learning Agile Robotic Locomotion Skills by Imitating Animals. In *Proceedings of Robotics: Science and Systems*.
- Xue Bin Peng, Ze Ma, Pieter Abbeel, Sergey Levine, and Angjoo Kanazawa. 2021. AMP: Adversarial motion priors for stylized physics-based character control. *ACM Transactions on Graphics (TOG)* 40, 4, Article 144 (2021).
- Ivaylo Popov, Nicolas Heess, Timothy Lillicrap, Roland Hafner, Gabriel Barth-Maron, Matej Vecerik, Thomas Lampe, Yuval Tassa, Tom Erez, and Martin Riedmiller. 2017. Data-efficient deep reinforcement learning for dexterous manipulation. *arXiv preprint arXiv:1704.03073* (2017).
- Madhu Ragupathi, Diego I Ramos-Valadez, Rodrigo Pedraza, and Eric M Haas. 2010. Robotic-assisted single-incision laparoscopic partial colectomy. *The International Journal of Medical Robotics and Computer Assisted Surgery* 6, 3 (2010), 362–367.
- Akshara Rai, Rika Antonova, Seungmoon Song, William Martin, Hartmut Geyer, and Christopher Atkeson. 2018. Bayesian optimization using domain knowledge on the ATRIAS biped. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 1771–1778.
- Aravind Rajeswaran, Vikash Kumar, Abhishek Gupta, Giulia Vezzani, John Schulman, Emanuel Todorov, and Sergey Levine. 2018. Learning Complex Dexterous Manipulation with Deep Reinforcement Learning and Demonstrations. In *Proceedings of Robotics: Science and Systems*.
- Ahmed A Ramadan, Tomohito Takubo, Yasushi Mae, Kenichi Oohara, and Tatsuo Arai. 2009. Developmental process of a chopstick-like hybrid-structure two-fingered micromanipulator hand for 3-D manipulation of microscopic objects. *IEEE Transactions on Industrial Electronics* 56, 4 (2009), 1121–1135.
- Carl Edward Rasmussen. 2003. Gaussian processes in machine learning. In *Summer School on Machine Learning*. Springer, 63–71.
- Haruka Sakurai, Takahiro Kanno, and Kenji Kawashima. 2016. Thin-diameter chopsticks robot for laparoscopic surgery. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 4122–4127.
- John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. 2015. Trust region policy optimization. In *International conference on machine learning*. PMLR, 1889–1897.
- John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347* (2017).
- Niranjan Srinivas, Andreas Krause, Sham M Kakade, and Matthias Seeger. 2010. Gaussian process optimization in the bandit setting: No regret and experimental design. In *Proceedings of the 27th annual international conference on machine learning*. 1015–1022.
- Omid Taheri, Nima Ghorbani, Michael J Black, and Dimitrios Tzionas. 2020. GRAB: A dataset of whole-body human grasping of objects. In *European conference on computer vision (ECCV)*. Springer, 581–600.
- Jie Tan, Tingnan Zhang, Erwin Coumans, Atil Işcen, Yunfei Bai, Danijar Hafner, Steven Bohez, and Vincent Vanhoucke. 2018. Sim-to-Real: Learning Agile Locomotion For Quadruped Robots. In *Proceedings of Robotics: Science and Systems*.
- Emanuel Todorov. 2014. Convex and analytically-invertible dynamics with contacts and constraints: Theory and implementation in MuJoCo. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 6054–6061.
- Emanuel Todorov, Tom Erez, and Yuval Tassa. 2012. MuJoCo: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 5026–5033.
- Deepak Tolani and Norman I. Badler. 1996. Real-time inverse kinematics of the human arm. *Presence: Teleoperators and Virtual Environments* 5, 4 (1996), 393–401.
- Marc A Toussaint, Kelsey Rebecca Allen, Kevin A Smith, and Joshua B Tenenbaum. 2018. Differentiable physics and stable modes for tool-use and manipulation planning. (2018).
- Yangang Wang, Jianyuan Min, Jianjie Zhang, Yebin Liu, Feng Xu, Qionghai Dai, and Jinxiang Chai. 2013. Video-based hand manipulation capture through composite motion control. *ACM Transactions on Graphics (TOG)* 32, 4, Article 43 (2013).
- Nkengé Wheatland, Yingying Wang, Huaguang Song, Michael Neff, Victor Zordan, and Sophie Jörg. 2015. State of the art in hand and finger modeling and animation. In *Computer Graphics Forum*, Vol. 34. Wiley Online Library, 735–760.
- Jungdam Won and Jehee Lee. 2019. Learning body shape variation in physics-based characters. *ACM Transactions on Graphics (TOG)* 38, 6, Article 207 (2019).
- Yilin Wu, Wilson Yan, Thanard Kurutach, Larel Pinto, and Pieter Abbeel. 2019. Learning to manipulate deformable objects without demonstrations. *arXiv preprint arXiv:1910.13439* (2019).
- Kazuki Yamakawa, Yuko Tashiro, Mizuki Nakajima, and Tsuyoshi Saitoh. 2018. Development of a support system for holding chopsticks correctly. In *2018 International*

- Workshop on Advanced Image Technology (WAIT). IEEE, 1–2.
- Tomoko Yamauchi, Atsumi Koide, Atsuko Yamamoto, and Kazuko Oba. 2010. Effect of parental training on table manners and the way of holding chopsticks. *Journal of Cookery Science of Japan* 43, 4 (2010), 260–264.
- Akira Yamazaki and Ryosuke Masuda. 2012. Autonomous foods handling by chopsticks for meal assistant robot. In *ROBOTIK 2012; 7th German Conference on Robotics*. VDE, 1–6.
- Zeshi Yang and Zhiqi Yin. 2021. Efficient hyperparameter optimization for physics-based character animation. *Proceedings of the ACM on Computer Graphics and Interactive Techniques* 4, 1 (2021), 1–19.
- Yuting Ye and C Karen Liu. 2012. Synthesis of detailed hand manipulations using contact sampling. *ACM Transactions on Graphics (TOG)* 31, 4, Article 41 (2012).
- Zhiqi Yin, Zeshi Yang, Michel Van de Panne, and Kangkang Yin. 2021. Discovering Diverse Athletic Jumping Strategies. *ACM Transactions on Graphics (TOG)* 40, 4, Article 91.
- Andy Zeng, Shuran Song, Johnny Lee, Alberto Rodriguez, and Thomas Funkhouser. 2020. Tossingbot: Learning to throw arbitrary objects with residual physics. *IEEE Transactions on Robotics* 36, 4 (2020), 1307–1319.
- Wenyuan Zeng, Wenjie Luo, Simon Suo, Abbas Sadat, Bin Yang, Sergio Casas, and Raquel Urtasun. 2019. End-to-end interpretable neural motion planner. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 8660–8669.
- He Zhang, Yuting Ye, Takaaki Shiratori, and Taku Komura. 2021. ManipNet: Neural manipulation synthesis with a hand-object spatial representation. *ACM Transactions on Graphics (TOG)* 40, 4, Article 121 (2021).
- Yunbo Zhang, Wenhao Yu, C Karen Liu, Charlie Kemp, and Greg Turk. 2020. Learning to manipulate amorphous materials. *ACM Transactions on Graphics (TOG)* 39, 6, Article 189 (2020).
- Wenping Zhao, Jianjie Zhang, Jianyuan Min, and Jinxiang Chai. 2013. Robust realtime physics-based motion control for human grasping. *ACM Transactions on Graphics (TOG)* 32, 6, Article 207 (2013).

A INPUT STATE OF THE HAND CONTROLLERS

We denote the input state of our low-level hand controllers as a vector s . It consists of multiple components as listed in Table 2. s_{hand} , s_{chop} , and s_{obj} are the simulation states of the hand and arm, the chopsticks, and the object, respectively. d_{hand} measures the distances between the fingertips and their desired contact locations on the chopsticks as specified in the gripping pose. f_{hand} and f_{chop} are the magnitude of the normal forces between the fingertips and the chopsticks, and between the chopsticks and the object, respectively.

Quantities with a tilde on top represent desired states in the planned trajectory to track. In particular, $\tilde{s}_{\text{chop}} = (\tilde{q}_{\text{chop}}, \dot{\tilde{q}}_{\text{chop}})$, where q_{chop} is parameterized as the 7-DoF parallel gripper as described in Section 6.1. To encourage smooth tracking, we include six frames of these desired states sampled every 0.05s for the next 0.3s.

Table 2. Components of the state s of our hand controllers.

symbol	description
s_{hand}	simulation state of the hand and arm
s_{chop}	simulation state of the chopsticks
s_{obj}	simulation state of the object
t_{obj}	shape parameters of the object
d_{hand}	distance between the fingertips and their desired contact locations on the chopsticks
f_{hand}	magnitude of contact forces on fingertips
f_{chop}	magnitude of contact forces on chopsticks tips
$\tilde{s}_{\text{hand}} \times 6$	planned states of the hand and arm
$\tilde{s}_{\text{chop}} \times 6$	planned states of the chopsticks
$\tilde{s}_{\text{obj}} \times 6$	planned states of the object

B ADDITIONAL OPTIMIZED GRIPPING POSES

Optimized gripping poses are found by Bayesian Optimization and DRL for seventeen valid gripping styles. In the main text we show the five most distinctive styles, and here we show the remaining twelve styles in Figure 22.

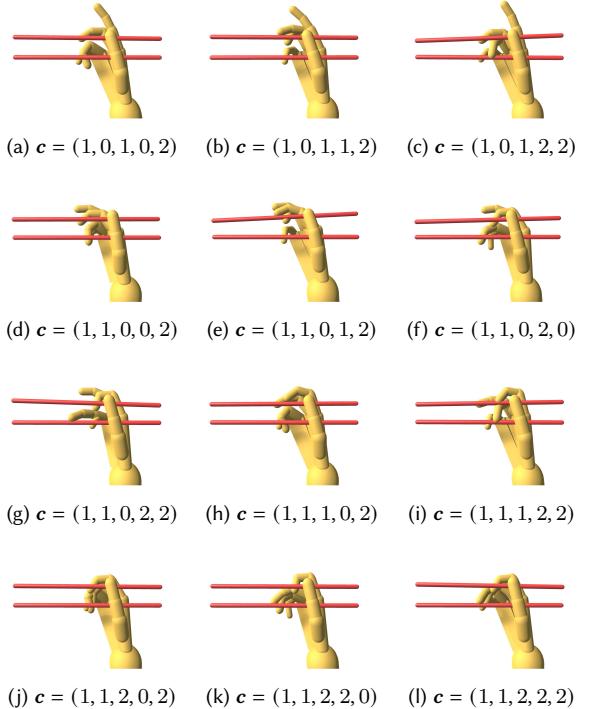


Fig. 22. Visualization of twelve additional optimized gripping poses.