Benchmarking Structured Policies and Policy Optimization for Real-World Dexterous Object Manipulation

Niklas Funk*[1], Charles Schaff*[2], Rishabh Madan*[3], Takuma Yoneda*[2], Julen Urain De Jesus[1]
Joe Watson[1], Ethan K. Gordon[4], Felix Widmaier[5], Stefan Bauer[5], Siddhartha S. Srinivasa[4]
Tapomayukh Bhattacharjee[4], Matthew R. Walter[2], Jan Peters[1]

***Abstract*— Dexterous manipulation is a challenging and important problem in robotics. While data-driven methods are a promising approach, current benchmarks require simulation or extensive engineering support due to the sample inefficiency of popular methods. We present benchmarks for the TriFinger system, an open-source robotic platform for dexterous manipulation and the focus of the 2020 Real Robot Challenge. The benchmarked methods, which were successful in the challenge, can be generally described as *structured policies*, as they combine elements of classical robotics and modern policy optimization. This inclusion of inductive biases facilitates sample efficiency, interpretability, reliability and high performance. The key aspects of this benchmarking is validation of the baselines across both simulation and the real system, thorough ablation study over the core features of each solution, and a retrospective analysis of the challenge as a manipulation benchmark. The code and demo videos for this work can be found on our website (`https://sites.google.com/view/benchmark-rrc`).**

I. INTRODUCTION

Dexterous manipulation is a challenging problem in robotics, that has impactful applications across industrial and domestic settings. Manipulation is challenging due to a combination of environment interaction, high-dimensional control and required exteroception. As a consequence, designing high-performance control algorithms for physical systems remains a challenge.

Due to the complexity of the problem, data-driven approaches to dexterous manipulation are a promising direction. However, due to the high cost of collecting data with a physical manipulator and the sample efficiency of current methods, the robot learning community has primarily focused on simulated experiments and benchmarks [1, 2, 3, 4, 5]. While there have been successes on hardware for various manipulation tasks [6, 7, 8, 9], the hardware and engineering cost of reproducing these experiments can be prohibitive to most researchers.

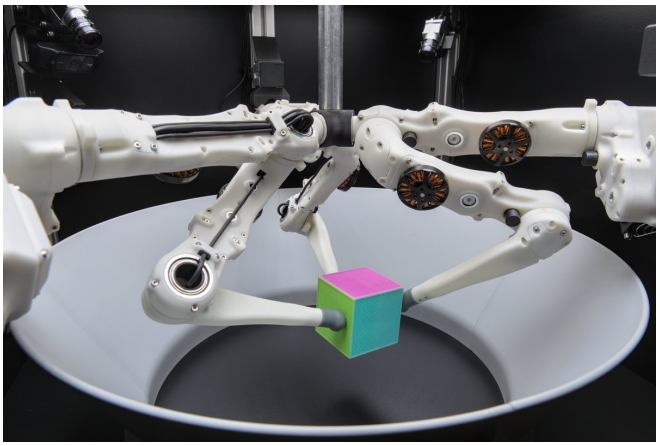

Fig. 1. The TriFinger platform performing a task in the Real Robot Challenge, bringing a cube object to a desired pose.

In this work, we investigate several approaches to dexterous manipulation using the TriFinger platform [10], an open-source manipulation robot. This research was motivated by the 'Real Robot Challenge' (RRC)[1], where the community was tasked with designing manipulation agents on a farm of physical TriFinger systems. A common theme in the successful solutions can be described as *structured policies*, methods that combine elements of classical robotics and modern machine learning to achieve reliability, sample efficiency and high performance. We summarize the solutions here, and analyse their performance through ablation studies to understand which aspects are important for real-world manipulation and how these characteristics can be appropriately benchmarked.

The main contributions for our work are as follows. We introduce **three independent structured policies** for trifinger object manipulation and **two data-driven optimization schemes**. We perform a detailed **benchmarking and ablation study** across policy structures and optimization schemes, with evaluations both in simulation and on several TriFinger robots. The paper is structured as follows: Section II discusses prior work, Section III introduces the TriFin-

*Equal contribution. Names are displayed in a random order.

[1]Department of Computer Science, Technical University of Darmstadt, {niklas, julen, joe, jan}@robot-learning.de
[2]Toyota Technological Institute at Chicago, Chicago IL, USA, {cbschaff, takuma, mwalter}@ttic.edu
[3]SAGA Robotics, Lincoln, UK, rishabhmadan96@gmail.com
[4]University of Washington, Seattle WA, USA, {ekgordon, siddh, tapo}@uw.edu
[5]Max Planck Institute for Intelligent Systems, Tübingen, Germany, {felix.widmaier, stefan.bauer}@tue.mpg.de

[1]See `https://real-robot-challenge.com`.

ger platform, Section IV describes the structured policies, Section V presents the approaches to structured policy optimization, and Section VI details the experiments. Finally, Section VII discusses the findings.

II. RELATED WORK

Much progress has been made for manipulation in structured environments with a priori known object models through the use of task-specific methods and programmed motions. However, these approaches typically fail when the environment exhibits limited structure or is not known exactly. The contact-rich nature of manipulation tasks naturally demands compliant motion. This has led to the development of control objectives like force control [11], hybrid position/force control [12], and impedance control [13]. Operational space control [14] has particularly been monumental in developing compliant task space controllers that use torque control.

Data-driven Manipulation Given the complexity of object manipulation due to both the hardware and task, data-driven approaches are an attractive means to avoid the need to hand-design controllers, while also providing potential for improved generalizability and robustness. For control, a common approach is to combine learned models with optimal control, such as guided policy search [15] and MPC [9, 16]. Model-free reinforcement learning (RL) has also been applied to manipulation [17, 18], including deep RL [5, 7], which typically requires demonstrations or simulation-based training due to sample complexity. Data-driven methods can also improve grasp synthesis [19, 20].

Structured Policies Across machine learning, *inductive biases* are means of introducing domain knowledge to improve sample efficiency, interpretability and reliability. In the context of control, inductive biases have been applied to enhance models for model-based RL [21], or to policies to simplify or improve policy search. Popular structures include options [22], dynamic movement primitives [23, 24, 25], autoregressive models [26], model predictive control [27] and motion planners [28, 29]. Structure also applies to the action representation, as acting in operational space, joint space or pure torque control affects how much classical robotics can be incorporated into the complete control scheme [30].

Residual Policy Learning Residual policy learning [31, 32] provides a way to enhance a given base control policy by learning additive, corrective actions using RL [33]. This allows well developed tools in robotics—such as motion planning, PID controllers, etc.—to be used as an inductive bias for learned policies in a RL setting. This combination improves sample efficiency and exploration, leading to policies that can outperform classical approaches and pure RL.

Bayesian Optimization for Control Bayesian optimization (BO) [34] is a sample-efficient black-box optimization method that leverages the epistemic uncertainty of a Gaussian process model of the objective function to guide optimization. Due to this property, it can be used for hyperparameter optimization, policy search, sim-to-real transfer and grasp selection [34, 35, 36, 20].

Strategy	Level 1	Level 2	Level 3	Level 4	Total
MP-PG	-5472	-2898	-9080	-21428	-124221
CPC-TG	-3927	-4144	-4226	-48572	-219182
CIC-CG	-6278	-13738	-17927	-49491	-285500

TABLE I

Final results of the RRC Phase 2. The score of each level is the average reward over multiple runs. *Total* is a weighted sum returns over the four levels. The strategies of our independent submissions to the challenge are: Grasp and Motion planning (MP-PG), Cartesian position control with Triangle Grasp (CPC-TG), and Cartesian impedance control with Center of Three Grasp(CIC-TG). They ranked 1st, 2nd and 4th in the competition.

III. TRI-FINGER OBJECT MANIPULATION

In 2020, Wüthrich et al. introduced the TriFinger robot [10], shown in Fig. 1. The inexpensive and open source platform can be easily recreated and serves as the basis to the RRC which aims to promote state-of-the-art research in dexterous manipulation on real hardware.

A. About the Robot

The TriFinger robot consists of three identical "fingers" with three degrees of freedom each. Its robust design together with several safety measures allows running learning algorithms directly on the real robot, even if they send unpredictable or random commands. The robot can be controlled with either torque or position commands at a rate of 1 kHz. It provides measurements of angles, velocities and torques of all joints at the same rate. A vision-based tracking system provides the pose of the manipulated object with a frequency of 10 Hz. Users can interact with the platform by submitting experiments and downloading the logged data.

B. The Real Robot Challenge

A cluster of seven TriFinger robots served as the basis to the *Real Robot Challenge 2020*. While the herein presented methods have been used in all the three phases of the challenge, in the following, we focus on the second phase.

This phase deals with the task of moving a cube (shown in Fig. 1) from the center of the arena to a desired goal position. The cube has a side-length of 65 mm and weights about 94 g. The surface of the cube is structured to make grasping easier and each side has a different color to help vision-based pose estimation.

The phase is subdivided into four difficulty levels (L1-L4). We focus on the final two, **L3** and **L4** which sample the goal pose from anywhere in the workspace, and only L4 considers the goal orientation.

Thus, for L3 the reward r only reflects the position error of the cube. It is computed as a normalized weighted sum of the distance on the x/y-plane and the distance on the z-axis: $r_3 = -\left(\frac{1}{2} \cdot \frac{\|\boldsymbol{e}_{xy}\|}{d_{xy}} + \frac{1}{2} \cdot \frac{|e_z|}{d_z}\right)$, where $\boldsymbol{e} = (e_x, e_y, e_z)$ is the error between actual and goal position, d_{xy} the range on the x/y-plane and d_z the range on the z-axis.

For L4, the orientation error is computed as the normalized magnitude of the rotation $\boldsymbol{q}$ (given as quaternion) between

actual and goal orientation $\text{err}_{\text{rot}} = 2\,\text{atan2}(||\boldsymbol{q}_{xyz}||, |q_w|)/\pi$ and the reward is $r_4 = (r_3 - \text{err}_{\text{rot}})/2$.

This paper benchmarks the solutions of three independent submissions to the challenge. Table I shows their performance.

IV. STRUCTURED POLICIES

This section describes the structured controllers considered as baselines. The controllers share a similar high-level structure and can be broken into three main components: cube alignment, grasping, and movement to the goal pose. We will discuss the individual grasp and goal approach strategies, and then briefly mention cube alignment. For a visualization of each grasping strategy, see Fig. 2. For a more detailed discussion of each controller and alignment strategy, please see the reports submitted for the RRC competition: motion planning [37], Cartesian position control [38], and Cartesian impedance control [39].

A. Grasp and Motion Planning (MP)

When attempting to manipulate an object to a desired goal pose, the grasp must be carefully selected such that it can be maintained throughout the manipulation. Many grasps which are valid at the current object pose may fail when moving the object. To avoid using such a grasp, we consider several heuristic grasps and attempt to plan a path from the initial object pose to the goal pose under the constraint that the grasp can be maintained at all points along the path. Path planning involves first selecting a potential grasp and then running a rapidly exploring random tree (RRT) [40] to generate a plan in task space, using the grasp to determine the fingers' joint positions. If planning is unsuccessful within a given time frame, we select another grasp and retry. We consider two sets of heuristic grasps, one with fingertips placed at the center of three vertical faces and the other with two fingers on one face and one on the opposite face. In the unlikely event that none of those heuristic grasps admits a path to the goal, we sample random force closure grasps [41] until a plan is found. Throughout this paper we refer to this method for determining a grasp as the **Planned Grasp (PG)**.

To move the object to the goal pose, this approach then simply executes the motion plan by following its waypoints in open loop using a PD position controller. After execution, to account for errors such as slippage or an inacurate final pose, we iteratively append waypoints to the plan in a straight line path from the current object position to the goal position.

B. Cartesian Position Control (CPC)

Much of the tasks presented in the challenge can be solved using carefully designed position-based motion primitives. We build upon this intution and implement a controller that uses Cartesian space position as reference and outputs joint torques to actuate the robot. To do so, we first reduce the tri-finger joint space $\boldsymbol{q} \in \mathbb{R}^9$ into the 3D Cartesian position space of each end effector $\boldsymbol{x} \in \mathbb{R}^9$, discarding finger orientation due to the rotational near-symmetry of the end effectors.

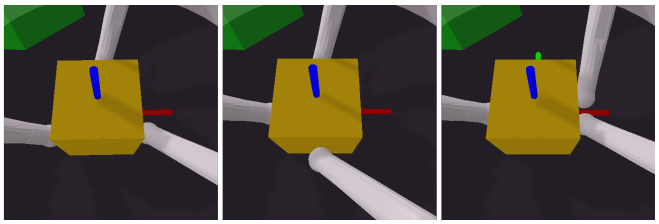

Fig. 2. Depicting the different grasp strategies. From left to right: triangle grasp (TG), center of three grasp (CG), and opposite faces grasp (OG). All permutations of fingers and vertical faces for the CGs and OGs are considered for the planned grasp (PG) heuristic.

We retrieve the Jacobian matrix, $\boldsymbol{J} := \frac{\partial \boldsymbol{x}}{\partial \boldsymbol{q}}$. The Jacobian-inverse converts the task velocity to joint velocity according to $\dot{\boldsymbol{q}} = \boldsymbol{J}^{-1}\dot{\boldsymbol{x}}$. $\boldsymbol{J}$ may be singular or non-square, so we use its damped pseudo inverse to guarantee a good solution at the cost of slight bias: $\dot{\boldsymbol{q}} = \boldsymbol{J}^\top \left(\boldsymbol{J}\boldsymbol{J}^\top + \lambda \boldsymbol{I}\right)^{-1} \dot{\boldsymbol{x}}$. We combine this with gravity compensation torques to command gravity-independent linear forces at each finger tip.

We build upon these linear force commands to create position-based motion primitives. Given a target position for each fingertip, we construct a feedback controller with PID gains tuned using [42] coupled with some minor adjustments in response to performance changes. This approach works well in simulation, but for the real system it results in fingers getting stuck in intermediate positions in some cases. Based on the limited interaction due to remote access to the robots, this could be attributed to static friction causing the motors to stop. We use a simple gain scheduling mechanism that varies the gains exponentially (up to a clipping value) over a specified time interval, which helps in mitigating this degradation in performance by providing the extra force required to keep the motors in motion.

Triangle Grasp (TG) The above controller is combined with this grasp which places fingers on three of the faces perpendicular to the ground. The fingers are placed such that they form an equilateral triangle [43]. This ensures that the object is in force closure and can easily apply forces on the center of mass of the cube in any direction.

C. Cartesian Impedance Control (CIC)

Third, we present a Cartesian impedance controller (CIC) [41, 44, 45, 46]. Using CIC enables natural adaptivity with the environment for object manipulation by specifying second-order impedance dynamics. This avoids having to learn the grasping behaviour through extensive experience and eludes complex trajectory optimization that must incorporate contact forces and geometry. Avoiding such complexity results in a controller that has adequate baseline performance on the real system that can then be further optimized.

For the desired cartesian position of the ith fingertip $\boldsymbol{x}_i$, we define $\bar{\boldsymbol{x}}_i$ to be the error between this tip position and a reference position inside the cube $\boldsymbol{x}_r$, so $\bar{\boldsymbol{x}}_i = \boldsymbol{x}_r - \boldsymbol{x}_i$. We then define an impedance controller for $\bar{\boldsymbol{x}}_i$, a second order ODE that can be easily interpreted as a mass

spring damper system with parameters $\boldsymbol{M}, \boldsymbol{D}, \boldsymbol{K}$. Since the cube's position is estimated using a vision-based system, the damping factor was zeroed for fast control. Converting this cartesian space control law back to joint coordinates results in $\boldsymbol{\tau}_{1,i} = \boldsymbol{M}(\boldsymbol{q})\boldsymbol{J}^{-1}\ddot{\boldsymbol{x}}_i$, where $\boldsymbol{\tau}_{1,i}$ denotes the torques to be applied to finger i.

To perform cube position control, we follow the ideas presented in [46] and design a proportional control law that perturbs the center of cube $\boldsymbol{x}_c$ based on the goal position $\boldsymbol{x}_g$, $\hat{\boldsymbol{x}}_r = \boldsymbol{x}_r + \boldsymbol{K}_1(\boldsymbol{x}_g - \boldsymbol{x}_c)$.

The above components do not consider the fingers as a whole, and so was limited in controlling the orientation of the cube. Contact forces were also passively applied rather than explicitly considered. To incorporate these additional considerations, we superimpose four torques. First is the already introduced position control and gravity compensation, which is added with three contact and rotational terms explained in the following, such that $\boldsymbol{\tau}_i = \sum_{j=1}^{4} \boldsymbol{\tau}_{j,i}$.

To also allow directly specifying the force applied by each finger, we introduce an additional component $\boldsymbol{\tau}_{2,i} = \boldsymbol{J}^\top \mathbf{F}_{2,i}$, where $\mathbf{F}_{2,i}$ is the force applied by finger i. We chose $\mathbf{F}_{2,i}$ to be in the direction of the surface normal of the face where finger i touches the cube ($\mathbf{F}_{2,i} = \boldsymbol{K}_2\boldsymbol{d}_i$). However, to not counteract the impedance controller, the resulting force of this component $\mathbf{F}_{\text{res}} = \sum_i \mathbf{F}_{2,i}$ should account to zero.

By solving $-\mathbf{F}_{\text{res}} = [\boldsymbol{J}^{-\top}, \boldsymbol{J}^{-\top}, \boldsymbol{J}^{-\top}][\boldsymbol{\tau}_{3,1}, \boldsymbol{\tau}_{3,2}, \boldsymbol{\tau}_{3,3}]^\top$ for $\boldsymbol{\tau}_{3,i}$, this is ensured. All previous components ensure a stable grasp closure. This is essential for the following orientation control law. Neglecting the exact shape of the cube, we model the moment that is exerted onto the cube as $\Omega = \sum \boldsymbol{r}_i \times \mathbf{F}_{4,i} = \sum \boldsymbol{S}_{r_i}\mathbf{F}_{4,i}$, where $\boldsymbol{r}_i = -\bar{\boldsymbol{x}}_i/|\bar{\boldsymbol{x}}_i|_2$ denotes the vector pointing from the center of cube towards the finger position, $\boldsymbol{S}_{r_i}$ the respective skew-symmetric matrix, and $\mathbf{F}_{4,i}$ an additional force that should lead to the desired rotation. The goal is now to realize a moment proportional to the current rotation errors, which are provided in the form of an axis of rotation $\boldsymbol{r}_\phi$ and its magnitude ϕ. Thus, the control law yields $\Omega = \boldsymbol{K}_3\phi\boldsymbol{r}_\phi$. We achieve Ω by solving $\Omega = [\boldsymbol{S}_{\boldsymbol{r}_1}\boldsymbol{J}^{-\top}, \boldsymbol{S}_{\boldsymbol{r}_2}\boldsymbol{J}^{-\top}, \boldsymbol{S}_{\boldsymbol{r}_3}\boldsymbol{J}^{-\top}][\boldsymbol{\tau}_{4,1}, \boldsymbol{\tau}_{4,2}, \boldsymbol{\tau}_{4,3}]^\top$, for $\boldsymbol{\tau}_{4,i}$.

Center of Three Grasp (CG) The above controller is combined with this grasp that places the fingers in the center of three out of the four faces that are perpendicular to the ground plane. The three faces are selected based on the task and goal pose with respect to the current object pose. When only a goal position is specified (L1-L3) the face which is closest to the goal location is not assigned any finger, ensuring that the cube can be pushed to the target. For L4 cube pose control, two fingers are placed on opposite faces such that the line connecting them is close to the axis of rotation. To avoid colliding with the ground, the third finger is placed such that an upward movement will yield the desired rotation.

D. Cube Alignment

While we will focus our experiments on the above methods, another component was necessary for the teams to perform well in the competition. Moving an object to an arbitrary pose often requires multiple interactions with the object itself. Therefore, teams had to perform some initial alignment of the cube with the goal pose. All three teams independently converged on a sequence of scripted motion primitives to achieve this. These primitives consisted of heuristic grasps and movements to (1) slide the cube to the center of the workspace, (2) perform a 90 degree rotation to change the upward face of the cube, (3) and perform a yaw rotation. Following this sequence, the cube is grasped and moved to the goal pose. For details on the specifics of primitives used by each team and their sequencing, please see their reports ([37, 38, 39]).

V. POLICY OPTIMIZATION

In addition to the approaches described above, the teams experimented with two different optimization schemes: Bayesian optimization and residual policy learning. In this section we will briefly introduce these methods.

A. Bayesian Optimization

BO is a black-box method to find the global optimum of an unknown function $f : \mathcal{X} \to \mathbb{R}$. This is achieved by approximating f with a nonparametric Gaussian process model $\mathcal{GP}(0, k)$. Our Gaussian processes provide a zero-mean prior distribution over functions with explicit uncertainty estimates in which prior knowledge about f is encoded through the covariance function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$. Using this prior, BO selects the next point $\boldsymbol{x} \in \mathcal{X}$ from a set of candidates such that $\boldsymbol{x}$ maximizes some criterion called the acquisition function. $f(\boldsymbol{x})$ is then evaluated and $\mathcal{GP}$ is updated with the result. Candidates are iteratively selected and evaluated until a budget constraint has been met.

In this work, we make use of the `BoTorch` optimization library [47] and use Bayesian optimization to improve the hyperparameters $\boldsymbol{\theta}$ of the previously introduced structured control policies. This is achieved by setting $f{=}\mathbb{E}_{\boldsymbol{g}}\left[\mathcal{R}(\boldsymbol{\theta}, \boldsymbol{g})\right]$, where $\boldsymbol{g}$ represents the respective goal pose for the experiment and $\mathcal{R}$ is the objective function used in the RRC. The expectation over $\mathcal{R}$ is approximated by averaging over N experiments, either in simulation or on the real platform. Lastly, the uncertainty of the Gaussian process model is estimated through Matérn $5/2$ kernels and the best candidate hyperparameters are obtained by maximizing the expected improvement acquisition function after fitting the model to the collected data.

B. Residual Policy Learning

In RL [33], an agent interacts with a MDP defined by the tuple $\mathcal{M} = \{\mathcal{S}, \mathcal{A}, \mathcal{T}, r, \gamma\}$. At any point in time, the agent is in a state $\boldsymbol{s}_t \in \mathcal{S}$ and produces an action $\boldsymbol{a}_t \in \mathcal{A}$. The MDP then transtions to a new state $\boldsymbol{s}_{t+1} \in \mathcal{S}$ and the agent recevies a reward $r(\boldsymbol{s}_t, \boldsymbol{a}_t)$. The goal of the agent is to select a policy π mapping states to actions that maximizes the discounted sum of rewards: $\pi^*{=}\arg\max_\pi \ \mathbb{E}_\pi \left[\sum_{t=0}^{T} \gamma^t r(\boldsymbol{s}_t, \boldsymbol{a}_t)\right]$.

In robotics, it is often easy to create a controller that obtains reasonable performance on the desired task. Residual policy learning [31, 32] aims to leverage an existing controller π_0 and use RL to learn corrective or *residual* actions to that controller. The learned policy acts by adding an action $\boldsymbol{a}_t^r \sim \pi(\boldsymbol{s}_t, \boldsymbol{a}_t^0)$ to the action $\boldsymbol{a}_t^0 \sim \pi_0(\boldsymbol{s}_t)$ provided by the controller. From the agent perspective this is a standard RL setting with the state space and transition dynamics augmented by π_0: $\mathcal{M} = (\mathcal{S} \times \mathcal{A}, \mathcal{A}, \mathcal{T}', \nabla, \gamma)$, where $\mathcal{T}'([\boldsymbol{s}_t, \boldsymbol{a}_t^0], \boldsymbol{a}_t^r, [\boldsymbol{s}_{t+1}, \boldsymbol{a}_{t+1}^0]) = \mathcal{T}(\boldsymbol{s}_t, \boldsymbol{a}_t^0 + \boldsymbol{a}_t^r, \boldsymbol{s}_{t+1})\mathbb{P}(\boldsymbol{a}_{t+1}^0 | \pi_0(\boldsymbol{s}_{t+1}))$. Residual policy learning benefits from the inductive bias of the base controller π_0 which greatly improves exploration and sample efficiency.

In this work, we learn residual controllers on top of the three structured approaches defined above. To do this, we use soft actor-critic (SAC) [1], a robust RL algorithm for control in continuous action space based on the maximum entropy RL framework. SAC has been successfully used to train complex controllers in robotics [48].

VI. EXPERIMENTS

To provide a thorough benchmark of the above methods, we perform a series of detailed experiments and ablations in which we test the contribution of different components on L3 and L4 of the RRC.

Experiment Setup In our experiments, we will be comparing different combinations of grasp strategies, controllers, and optimization schemes in simulation and on the TriFinger platform. For each combination, we will report the reward and final pose error averaged over several trials as well as the fraction of the time the object is dropped. In simulation, we provide each method with the same initial object poses and goal poses. On the TriFinger platform, we cannot directly control the initial pose of the object, so we first move it to the center of the workspace and the assign the goal pose as a relative transformation of the initial pose. All methods are tested with the same set of relative goal transformations. To isolate the performance of the grasp strategies and controllers, we initialize the experiments so that no cube alignment primitives are required to solve the task.

Mix and Match The choice of grasp heuristic and control strategy are both crucial to success, but it is hard to know how much each component contributed individually. To test each piece in isolation we "mix and match" the three grasp heuristics with the three structured controllers and report the performance of all nine combinations. Each combination is tested on L4 and results are averaged over 15 trials. Further, the speed and accuracy of the initial cube alignments had a large impact on the competition reward. To account for this and to test the robustness of these approaches to different alignment errors, we evaluate each approach under three different initial orientation errors $\delta\theta \in \{10, 25, 35\}$ degrees. When the MP controller is paired with when a grasp strategy other than PG, a motion plan is generated using that grasp.

Bayesian Optimization All of our approaches are highly structured and rely on a few hyperparameters. We investigate whether BO can improve the performance of the controllers through optimizing them for both L3 and L4. For all experiments, we initialize the optimization with 4 randomly sampled intial sets of parameters and run 50 optimization iterations. We do not explicitly exploit any information from our manually tuned values. Instead, the user only has to specify intervals. This sample-efficient optimization process only lasts about 12 hours for optimizing on the real system. After the optimized parameters are found, each controller is tested over 20 trials on the TriFinger platform with both the manually tuned parameters and the BO parameters. For our proposed approaches, we optimize the following values: CIC - gains and reference position $\boldsymbol{x}_r$. CPC - gains, including values for the exponential gain scheduling. MP - hyperparameters that control the speed of the movement to the goal location on the planned path.

Residual Policy Learning In these experiments, we investigate to what exent residual policy learning can be used to improve the performance of our three controllers. To do this we train a neural network policy to produce joint torques in the bounded range $[-0.05, 0.05]$ (the maximum allowed joint torque on the system is 0.397), which are then added to the actions of the base controller. The policies are trained on L3 in simulation for 1M timesteps using SAC [1]. The reward function consists of a combination of the competition reward added with terms for action regularization, maximizing tip force sensor readings, and maintianing the grasp of the object. The policy architecture is as follows: The observation and base action are separately embedding into a 64 dimensional space. These embeddings are then concatenated and passed to a three-layer feed forward network which outputs a gaussian distribution over torque actions. Actions are sampled from this distribution, squashed using a Tanh function, then scaled to fit in the [-0.05, 0.05] range. We evaluate MP-PG, CPC-TG, and CIC-CG in simulation over 20 trials for L3. We then test the ability of these policies to transfer to the real system with another 20 trials.

VII. RESULTS

In this section we describe the results from running the experiments described above. In total, we conducted more than 20k experiments on the real system.

Mix and Match Experiment results on the real platform are summarized in Table II. *Drop* refers to the percentage of the episodes that the cube is dropped. *Pos.* and *Ori. Err.* are the position and orientation errors at the end of the episodes. The values are only calculated from those runs that do not drop the cube. As a common trend, orientation error increases as $\delta\theta$ gets bigger, and except for CPC, position errors stay close to zero with low variance across different settings.

We find that CPC drops the cube more frequently than other approaches, and its performance varies a lot depending on the choice of grasp. This is because it drives the fingertip positions to the goal pose without considering whether the grasp can be maintained. However, in the cases in which it can retain its grasp, it achieves much lower orientation errors than CIC or MP. While CIC is similar to CPC, it is

Controller	Grasp	$\delta\theta = 10$ deg Ori. Err. [deg]	$\delta\theta = 10$ deg Pos. Err. [cm]	$\delta\theta = 10$ deg Drop [%]	$\delta\theta = 25$ deg Ori. Err. [deg]	$\delta\theta = 25$ deg Pos. Err. [cm]	$\delta\theta = 25$ deg Drop [%]	$\delta\theta = 35$ deg Ori. Err. [deg]	$\delta\theta = 35$ deg Pos. Err. [cm]	$\delta\theta = 35$ deg Drop [%]
CIC	CG	55.9 ± 27.37	0.3 ± 0.14	0.0	$\mathbf{47.0 \pm 29.64}$	$\mathbf{0.8 \pm 1.12}$	**0.0**	58.2 ± 20.95	0.4 ± 0.16	6.7
	PG	51.3 ± 26.72	0.4 ± 0.60	0.0	53.6 ± 27.26	0.7 ± 0.96	0.0	$\mathbf{62.4 \pm 31.69}$	$\mathbf{0.4 \pm 0.24}$	**0.0**
	TG	$\mathbf{30.3 \pm 12.83}$	$\mathbf{0.4 \pm 0.22}$	**0.0**	39.4 ± 20.38	0.4 ± 0.17	13.3	30.7 ± 17.37	0.4 ± 0.21	6.7
CPC	CG	27.6 ± 36.74	3.0 ± 4.61	26.7	8.6 ± 5.24	1.5 ± 3.16	33.3	17.2 ± 15.38	3.3 ± 5.26	60.0
	PG	14.0 ± 16.98	2.6 ± 4.65	26.7	36.2 ± 39.84	4.6 ± 4.49	33.3	$\mathbf{35.4 \pm 45.88}$	$\mathbf{6.3 \pm 6.14}$	**13.3**
	TG	$\mathbf{5.0 \pm 3.19}$	$\mathbf{0.5 \pm 0.26}$	**6.7**	$\mathbf{20.3 \pm 48.96}$	$\mathbf{1.5 \pm 3.26}$	**26.7**	5.0 ± 2.99	0.4 ± 0.19	33.3
MP	CG	28.7 ± 10.86	0.4 ± 0.43	0.0	29.0 ± 13.95	0.5 ± 0.47	0.0	43.6 ± 11.49	0.6 ± 0.61	0.0
	PG	30.5 ± 14.56	0.5 ± 0.46	0.0	$\mathbf{28.0 \pm 8.70}$	$\mathbf{0.7 \pm 0.60}$	**0.0**	$\mathbf{30.5 \pm 10.85}$	$\mathbf{0.6 \pm 0.80}$	**0.0**
	TG	$\mathbf{26.7 \pm 11.24}$	$\mathbf{0.4 \pm 0.51}$	**0.0**	32.8 ± 15.22	0.6 ± 0.42	0.0	37.1 ± 13.31	0.6 ± 0.64	0.0

TABLE II

Mix-and-match experiment on the real platform. $\delta\theta$ denotes the error between the goal and initial cube orientation, *Drop* is the fraction of episodes that the cube is dropped. *Pos. Err.* and *Ori. Err.* are the position and orientation errors at the end of the episodes. Each value is calculated from 15 episodes.

Control Policy	L3 Real System $\mathcal{R}$	L3 Real System Pos. Err. [cm]	L3 Real System Ori. Err. [deg]	L4 Real System $\mathcal{R}$	L4 Real System Pos. Err. [cm]	L4 Real System Ori. Err. [deg]
CIC-CG	-7818.9 ± 3332.2	4.07 ± 2.49	N/A	-6998.1 ± 1840.5	0.86 ± 1.63	42.18 ± 27.40
CIC-CG w BO	$\mathbf{-5613.6 \pm 2643.7}$	$\mathbf{2.15 \pm 2.02}$	N/A	-7534.4 ± 2976.9	1.17 ± 2.55	48.59 ± 37.20
CPC-TG	-2912.2 ± 1738.0	1.43 ± 3.51	N/A	-6150.8 ± 2754.7	2.53 ± 4.13	27.06 ± 48.68
CPC-TG w BO	$\mathbf{-2130.5 \pm 1149.5}$	$\mathbf{0.48 \pm 1.04}$	N/A	$\mathbf{-5046.5 \pm 1664.7}$	$\mathbf{1.05 \pm 1.60}$	$\mathbf{13.95 \pm 24.76}$
MP-PG	-6267.3 ± 4363.8	2.15 ± 4.41	N/A	-7105.0 ± 2109.1	0.64 ± 0.11	25.62 ± 15.04
MP-PG w BO	$\mathbf{-4510.4 \pm 1412.4}$	$\mathbf{0.53 \pm 0.71}$	N/A	-7239.2 ± 2257.7	0.45 ± 0.66	22.39 ± 11.90

TABLE III

Comparing the manually obtained parameters with the ones obtained from running BO on the real system for the L3 (left) and L4 (right) experiments.

more robust to drops at the expense of accuracy because it explicitly considers the forces the fingertips need to apply in order to achieve a desired motion. MP is the most robust against drops and grasp choices because it attempts to move only to locations in which the selected grasp can be maintained. This comes at the expense of orientation errors because the planner may have only found a point near the goal for which the grasp is valid.

When comparing CG and TG, we find that TG performs better in terms of both orientation error and drop rate. We hypothesize that this is the case because the triangle shape facilitates to apply forces to the cube in all directions. The benefits of the planned grasp (PG) become apparent when the initial orientation error is large, improving the drop rate across all three controllers. This verifies the intuition for grasp planning that, when the required orientation change is large, it helps to carefully select a grasp that is feasible both at the initial pose and near the goal pose.

Bayesian Optimization Table III depicts the results from running BO on the real system to optimize the hyperaprameters for the controllers. As can be seen on the left hand side, for the L3 experiments, the newly obtained hyperparameters significantly improve the policies' mean reward as well as the the mean position errors. Furthermore, although during training, we only averaged across 5 rollouts, the improvements persist when evaluating the policies on 20 newly sampled goal locations. From visually inspecting the rollouts, we conclude that running BO results in higher gains such that the target locations are reached quicker.

Repeating the same experiment for L4 yields the results presented on the right hand side of Table III. As shown in the table, only the performance for the CPC control strategy can be improved significantly. Comparing the two sets of parameters, the BO algorithm suggests to use lower gain values. This results in a more stable and reliable control policy and increases performance. In general, even though we do not provide the manually obtained parameters as a prior to the BO algorithm, for the other two approaches, the performance of the optimized parameters is still on par with the manually tuned ones. We reason that for the MP approach, the two hyperparameters might not provide enough flexibility for further improvements while the CIC controller might have already reached its performance limits. The results indicate that BO is an effective tool to optimize and obtain performant hyperparameters for our approaches and mitigates the need for tedious manual tuning.

Residual Learning Table IV shows the performance of our control strategies with and without residual policy learning. We find that residual policy learning is only somewhat effective, improving only the MP controller. When inspecting the results we find that residual control is able to help the MP policy to maintain a tight grasp on the cube, preventing catastrophic errors such as dropping the cube and triggering another round of planning.

Surprisingly, the learned policies are able to transfer to the real system without any additional finetuning or algorithms such as domain randomization [49]. For the MP controller, we find improved grasp robustness and a reduction in the drop rate. Additionally for CPC and CIC, it is surprising that permormance was not more adversly affected given the

Control Policy	Simulation $\mathcal{R}$	Simulation Pos. Err. [cm]	Simulation Drop [%]	Transfer to the Real System $\mathcal{R}$	Transfer to the Real System Pos. Err. [cm]	Transfer to the Real System Drop [%]
CIC-CG	$\mathbf{-112 \pm 32.2}$	$\mathbf{0.55 \pm 0.28}$	**0.0**	-4410 ± 3304	1.60 ± 1.74	10.0
CIC-CG w RL	-570 ± 350.6	3.52 ± 3.91	10.0	-4593 ± 3964	1.54 ± 1.53	10.0
CPC-TG	-70.0 ± 9.42	0.25 ± 0.15	0.0	$\mathbf{-2150 \pm 860}$	$\mathbf{0.50 \pm 0.41}$	0.0
CPC-TG w RL	-74.4 ± 4.97	0.18 ± 0.12	0.0	-2847 ± 1512	1.46 ± 3.16	0.0
MP-PG	-122 ± 58.2	0.35 ± 0.43	0.0	-8638 ± 6197	8.47 ± 8.65	40.0
MP-PG w RL	$\mathbf{-85.8 \pm 22.6}$	$\mathbf{0.22 \pm 0.18}$	0.0	$\mathbf{-4573 \pm 1547}$	$\mathbf{1.74 \pm 3.10}$	**0.0**

TABLE IV

This table shows the results of combining our controllers with residual policy learning. Reward and final pose errors are shown for L3 in simulation and transferring the learned policies to the real system. Surprisingly, we find that in the case of the MP controller, the learned policy transfers to the real system without any additional work to help manage the domain shift.

lack of improvement in simulation. We hyothesize that this is the case for two reasons. One, the torque limits on the residual controller are small and may not be able to cause a collapse in performance, and two, the base controllers provide increasing commands as errors are made that work to keep the combined controller close to the training data distribution. For example, the MP controller has a predefined path and the PD controller following that path provides increasing commands as deviations occur.

From these experiments, we conclude that residual policy learning may be effective when small changes can be made, such as helping maintain contact forces, to improve the robustness of a controller, and that transferring a residual controller from simulation is substantially easier than transferring pure RL policies.

Challenge Retrospective Overall, the newly obtained results (see, e.g. Table III) differ only slightly compared to the scores reported from the competition in Table I. CPC-TG yields the best results on L3, but the team using MP-PG was able to win. In part because they implemented more reliable primitives for cube alignment. Through exploiting insights from this approach, we assume that CPC-TC could perform similarly. Nevertheless, the robustness of MP-PG might still outweigh the gains on the successful runs of the reactive policies, especially with increasing task difficulty.

VIII. CONCLUSION AND OUTLOOK

In this work, we present three different approaches to solving the tasks from the RRC. We perform extensive experiments in simulation and on the real platform to compare and benchmark the methods.

We find that using motion planning provides the best trade-off between accuracy and reliability. Compared to motion planning, the two reactive approaches vary in both reliability and accuracy. Concerning grasp selection, the results show that using the triangle grasp yields best performance.

We further show the effectiveness of running Bayesian optimization for hyperparameter optimization. Especially for L3, the performance can be increased significantly across all approaches. Augmenting the structured methods with a learned residual control policy can improve the performance when small changes to a controller are beneficial. Surprisingly, we also find that transferring the learned residual controllers required no finetuning on the real system or other techniques to cross the sim-to-real gap, although applying those techniques is likely to be beneficial.

We hope that our work serves as a benchmark for future competitions and dexterous manipulation research using the TriFinger platform.

ACKNOWLEDGMENT

This project has received funding from the European Union's Horizon 2020 program under grant agreement No. 640554 (SKILLS4ROBOTS). Niklas Funk acknowledges the support of the Nexplore/HOCHTIEF Collaboration Lab at TU Darmstadt. Charles Schaff and Takuma Yoneda were supported in part by the National Science Foundation under Grant No. 1830660. Research reported in this publication was supported by the Eunice Kennedy Shriver National Institute Of Child Health & Human Development of the National Institutes of Health under Award Number F32HD101192. The content is solely the responsibility of the authors and does not necessarily represent the official views of the National Institutes of Health. This work was also (partially) funded by the National Science Foundation IIS (#2007011), National Science Foundation DMS (#1839371), the Office of Naval Research, US Army Research Laboratory CCDC, Amazon, and Honda Research Institute USA.

REFERENCES

[1] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel *et al.*, "Soft actor-critic algorithms and applications," *arXiv preprint arXiv:1812.05905*, 2018.

[2] S. Fujimoto, H. van Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," *arXiv preprint arXiv:1802.09477*, 2018.

[3] I. Popov, N. Heess, T. Lillicrap, R. Hafner, G. Barth-Maron, M. Vecerik, T. Lampe, Y. Tassa, T. Erez, and M. Riedmiller, "Data-efficient deep reinforcement learning for dexterous manipulation," *arXiv preprint arXiv:1704.03073*, 2017.

[4] H. Charlesworth and G. Montana, "Solving challenging dexterous manipulation tasks with trajectory optimisation and reinforcement learning," *arXiv preprint arXiv:2009.05104*, 2020.

[5] A. Rajeswaran, V. Kumar, A. Gupta, G. Vezzani, J. Schulman, E. Todorov, and S. Levine, "Learning Complex Dexterous Manipulation with Deep Reinforcement Learning and Demonstrations," in *Proc. Robotics: Science and Systems (RSS)*, 2018.

[6] S. Gu, E. Holly, T. Lillicrap, and S. Levine, "Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates," in *Proc. IEEE Int'l Conf. on Robotics and Automation (ICRA)*, 2017.

[7] M. Andrychowicz, B. Baker, M. Chociej, R. Jozefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray, J. Schneider, S. Sidor, J. Tobin, P. Welinder, L. Weng, and W. Zaremba, "Learning dexterous in-hand manipulation," *The International Journal of Robotics Research*, 2020.

[8] I. Akkaya, M. Andrychowicz, M. Chociej, M. Litwin, B. McGrew, A. Petron, A. Paino, M. Plappert, G. Powell, R. Ribas *et al.*, "Solving rubik's cube with a robot hand," *arXiv preprint arXiv:1910.07113*, 2019.

[9] A. Nagabandi, K. Konoglie, S. Levine, and V. Kumar, "Deep Dynamics Models for Learning Dexterous Manipulation," in *Proc. Conf. on Robot Learning (CoRL)*, 2019.

[10] M. Wüthrich, F. Widmaier, F. Grimminger, J. Akpo, S. Joshi, V. Agrawal, B. Hammoud, M. Khadiv, M. Bogdanovic, V. Berenz *et al.*, "Trifinger: An open-source robot for learning dexterity," in *Proc. Conf. on Robot Learning (CoRL)*, 2020.

[11] M. T. Mason, "Compliance and force control for computer controlled manipulators," *IEEE Transactions on Systems, Man, and Cybernetics*, 1981.

[12] M. H. Raibert and J. J. Craig, "Hybrid position/force control of manipulators," 1981.

[13] N. Hogan, "Impedance control: An approach to manipulation: Part i—theory," 1985.

[14] O. Khatib, "A unified approach for motion and force control of robot manipulators: The operational space formulation," *IEEE Journal on Robotics and Automation*, 1987.

[15] V. Kumar, E. Todorov, and S. Levine, "Optimal control with learned local models: Application to dexterous manipulation," in *Proc. IEEE Int'l Conf. on Robotics and Automation (ICRA)*, 2016.

[16] K. Lowrey, A. Rajeswaran, S. Kakade, E. Todorov, and I. Mordatch, "Plan online, learn offline: Efficient learning and exploration via model-based control," in *International Conference on Learning Representations*, 2019.

[17] P. Falco, A. Attawia, M. Saveriano, and D. Lee, "On policy learning robust to irreversible events: An application to robotic in-hand manipulation," *IEEE Robotics and Automation Letters*, 2018.

[18] H. van Hoof, T. Hermans, G. Neumann, and J. Peters, "Learning robot in-hand manipulation with tactile features," in *Proc. Int'l Conf. on Humanoid Robots (HUMANOIDS)*, 2015.

[19] J. Bohg, A. Morales, T. Asfour, and D. Kragic, "Data-driven grasp synthesis—a survey," *IEEE Transactions on Robotics*, 2014.

[20] F. Veiga and A. Bernardino, "Towards bayesian grasp optimization with wrench space analysis," in *IEEE IROS 2012 Workshop "Beyond Robot Grasping"*, 2012.

[21] M. Lutter, J. Silberbauer, J. Watson, and J. Peters, "Differentiable physics models for real-world offline model-based reinforcement learning," in *Proc. IEEE Int'l Conf. on Robotics and Automation (ICRA)*, 2021.

[22] C. Daniel, H. van Hoof, J. Peters, and G. Neumann, "Probabilistic inference for determining options in reinforcement learning," 2016.

[23] A. Ijspeert, J. Nakanishi, and S. Schaal, "Learning attractor landscapes for learning motor primitives," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2003.

[24] A. Paraschos, C. Daniel, J. Peters, G. Neumann *et al.*, "Probabilistic movement primitives," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2013.

[25] S. Bahl, M. Mukadam, A. Gupta, and D. Pathak, "Neural dynamic policies for end-to-end sensorimotor learning," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[26] D. Korenkevych, A. R. Mahmood, G. Vasan, and J. Bergstra, "Autoregressive policies for continuous control deep reinforcement learning," *arXiv preprint arXiv:1903.11524*, 2019.

[27] S. Kamthe and M. Deisenroth, "Data-efficient reinforcement learning with probabilistic model predictive control," in *Proc. Int'l Conf. on Artificial Intelligence and Statistics (AISTATS)*, 2018.

[28] J. Yamada, Y. Lee, G. Salhotra, K. Pertsch, M. Pflueger, G. S. Sukhatme, J. J. Lim, and P. Englert, "Motion planner augmented reinforcement learning for obstructed environments," in *Proc. Conf. on Robot Learning (CoRL)*, 2020.

[29] F. Xia, C. Li, R. Martín-Martín, O. Litany, A. Toshev, and S. Savarese, "Relmogen: Leveraging motion generation in reinforcement learning for mobile manipulation," *arXiv preprint arXiv:2008.07792*, 2020.

[30] R. Martín-Martín, M. Lee, R. Gardner, S. Savarese, J. Bohg, and A. Garg, "Variable impedance control in end-effector space. an action space for reinforcement learning in contact rich tasks," in *Proc. IEEE/RSJ Int'l Conf. on Intelligent Robots and Systems (IROS)*, 2019.

[31] T. Silver, K. R. Allen, J. B. Tenenbaum, and L. P. Kaelbling, "Residual policy learning," *arXiv preprint arXiv:1812.06298*, 2018.

[32] T. Johannink, S. Bahl, A. Nair, J. Luo, A. Kumar, M. Loskyll, J. A. Ojea, E. Solowjow, and S. Levine, "Residual reinforcement learning for robot control," in *Proc. IEEE Int'l Conf. on Robotics and Automation (ICRA)*, 2019.

[33] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2018.

[34] J. Snoek, H. Larochelle, and R. P. Adams, "Practical bayesian optimization of machine learning algorithms," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2012.

[35] A. Marco, P. Hennig, J. Bohg, S. Schaal, and S. Trimpe, "Automatic lqr tuning based on gaussian process global optimization," in *Proc. IEEE Int'l Conf. on Robotics and Automation (ICRA)*, 2016.

[36] F. Muratore, C. Eilers, M. Gienger, and J. Peters, "Data-efficient domain randomization with bayesian optimization," *IEEE Robotics and Automation Letters*, 2021.

[37] T. Yoneda, C. Schaff, T. Maeda, and M. Walter, "Grasp and motion planning for dexterous manipulation for the real robot challenge," *arXiv preprint arXiv:2101.02842*, 2021.

[38] R. Madan, E. K. Gordon, T. Bhattacharjee, and S. S. Srinivasa, "Real robot challenge phase 2: Manipulating objects using high-level coordination of motion primitives," in *Submitted to Real Robot Challenge 2020*, 2020. [Online]. Available: https://openreview.net/forum?id=9tYX-lukeq

[39] Anonymous, "Model-based cartesian impedance control," in *Submitted to Real Robot Challenge 2020*, 2020. [Online]. Available: https://openreview.net/forum?id=JWUqwie0--W

[40] S. M. LaValle, "Rapidly-exploring random trees: A new tool for path planning," 1998.

[41] K. M. Lynch and F. C. Park, *Modern Robotics: Mechanics, Planning, and Control*, 2017.

[42] J. G. Ziegler, N. B. Nichols *et al.*, "Optimum settings for automatic controllers," *trans. ASME*, vol. 64, no. 11, 1942.

[43] M. A. Roa and R. Suárez, "Grasp quality measures: review and performance," *Autonomous robots*, 2015.

[44] T. Wimböck, C. Ott, A. Albu-Schäffer, and G. Hirzinger, "Comparison of object-level grasp controllers for dynamic dexterous manipulation," *Int'l J. of Robotics Research*, 2012.

[45] L. Biagiotti, H. Liu, G. Hirzinger, and C. Melchiorri, "Cartesian impedance control for dexterous manipulation," in *Proc. IEEE/RSJ Int'l Conf. on Intelligent Robots and Systems (IROS)*, 2003.

[46] M. Pfanne, M. Chalon, F. Stulp, H. Ritter, and A. Albu-Schäffer, "Object-level impedance control for dexterous in-hand manipulation," *IEEE Robotics and Automation Letters*, 2020.

[47] M. Balandat, B. Karrer, D. R. Jiang, S. Daulton, B. Letham, A. G. Wilson, and E. Bakshy, "BoTorch: A Framework for Efficient Monte-Carlo Bayesian Optimization," in *Advances*

in Neural Information Processing Systems (NeurIPS), 2020.

[48] T. Haarnoja, S. Ha, A. Zhou, J. Tan, G. Tucker, and S. Levine, "Learning to walk via deep reinforcement learning," *arXiv preprint arXiv:1812.11103*, 2018.

[49] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," in *Proc. IEEE/RSJ Int'l Conf. on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 23–30.