2024 IEEE International Conference on Robotics and Automation (ICRA 2024)
May 13-17, 2024. Yokohama, Japan

Dexterous In-hand Manipulation by Guiding Exploration with Simple Sub-skill Controllers

Gagan Khandate*†, Cameron Paul Mehlman*‡, Xingsheng Wei*‡, Matei Ciocarlie‡

***Abstract*— Recently, reinforcement learning has led to dexterous manipulation skills of increasing complexity. Nonetheless, learning these skills in simulation still exhibits poor sample-efficiency which stems from the fact these skills are learned from scratch without the benefit of any domain expertise. In this work, we aim to improve the sample efficiency of learning dexterous in-hand manipulation skills using controllers available via domain knowledge. To this end, we design simple sub-skill controllers and demonstrate improved sample efficiency using a framework that guides exploration toward relevant state space by following actions from these controllers. We are the first to demonstrate learning hard-to-explore finger-gaiting in-hand manipulation skills without the use of an exploratory reset distribution.**

***Index Terms*— Dexterous manipulation, Reinforcement learning, In-hand reorientation, In-hand manipulation, Guided exploration**

I. Introduction

Reinforcement learning has led to significant improvements in our ability to achieve a range of dexterous manipulation skills. Prior works have demonstrated in-hand manipulation skills such as object reorientation to a desired pose [1, 2] or continuous re-orientation about a given axis [3, 4]. However, learning these dexterous skills can still require up to billions of simulation steps.

Methods using domain expertise to improve the sample efficiency for learning in-hand manipulation skills achieve this mainly through engineered reset distributions and model-based controllers. The methods using reset distributions use a set of stable grasps to enable exploration which has been particularly effective for finger-gaiting in-hand manipulation tasks[3]. However, the reset distributions are object and task-specific and may require heavy engineering.

Alternatively, other methods use model-based controllers that leverage the model to ensure the stability of the grasp and combine them with reinforcement learning. Predominantly, this involves decomposing the in-hand manipulation task into appropriate sub-skills (i.e in-grasp manipulation, finger-pivoting, finger-gaiting, etc,.) and designing model-based controllers for each to be used as low-level controllers in hierarchical reinforcement learning [5].

However, this approach of using model-based controllers as low-level controllers has several limitations. First, from a training perspective, computing actions from these controllers can be expensive, particularly if it involves optimization. Hence, querying these controllers at every step can outweigh the benefits in terms of the net computational cost. Next, restricting the policy to only executing one of the pre-defined sub-skills can lead to learning a sub-optimal policy. Finally, a hierarchical learning framework implicitly requires that low-level model-based controllers must be deployable on hardware. Thus, these model-based controllers must be designed such that only sensor feedback available on the hardware is used as input, and controllers must be robust to uncertainties present on the hardware.

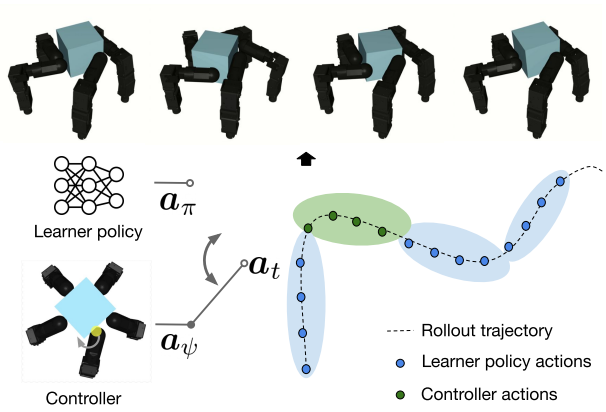


Fig. 1: Our method of interweaving the policy and sub-skill controller (ex. contact switching controller) during training allows the policy to effectively learn dexterous finger-gaiting skills as shown. Videos can be found at project page: roamlab.github.io/vge

In this work, we seek to leverage model-based controllers to improve sample efficiency while retaining the benefits of learning an end-to-end policy. The key idea that we propose is to interleave learner policies and sub-skill controllers during episode rollout during the initial phase and use only the learner policy during the later phase of training. Consequently, we use the sub-skill controllers only during training and not at run-time. We show that our method guides exploration towards relevant regions of the state-space. We also use the current action-value estimates provided by the critic to guide action selection which further improves exploration.

Critically, we use controllers that are simple to design and inexpensive to query actions from. Our controllers do not involve multi-step horizon motion planning or other sources of complexity and computational cost. This makes them inexpensive to compute - favorably reducing the cost of

* denotes joint first authorship
†Dept. of Computer Science, ‡Dept. of Mechanical Engineering
Columbia University, New York, NY 10027, USA
Corresponding email: gagank@cs.columbia.edu
This work was supported in part by the ONR under grant N00014-21-1-4010 and the NSF under grant CMMI-2037101.

querying the expert. However, inexpensive controllers will often be sub-optimal in their behavior. Still, our method enables learning despite using such sub-optimal controllers. Overall, our main contributions here include:

- We demonstrate that a set of simple sub-skill controllers can be leveraged to learn dexterous in-hand manipulation tasks such as finger-gaiting.
- In contrast to previous work combining model-based controllers and learning for manipulation, we use the proposed controllers for training an end-to-end policy that does not require these controllers for deployment. Our approach alleviates many constraints on the controllers we use, in terms of both performance and sensory input.
- We also show that sub-optimal controllers can enable effective exploration of the complex state space of our problem, without exploratory reset distributions used in previous studies with similar goals.

II. Related Work

Dexterous in-hand manipulation has long been of interest and numerous methods have been proposed ranging from simple low-level feedback primitive controllers to trajectory optimization for the complete task. Early model-based work on finger-gaiting [6–8] and finger-pivoting [9] propose controllers using simplified models. Fan et al. [10] and Sundaralingam et al. [11] use model-based online trajectory optimization and demonstrate finger-gaiting in simulation. Chen et al. [12] combine trajectory optimization and tree search to follow a desired object pose trajectory.

Recently, remarkable progress in dexterous manipulation has been achieved with model-free RL, typically, demonstrating reorientation to a desired object pose [1]. However, model-free RL requires hundreds of hours of training. It was shown that GPU physics could be used to accelerate learning these skills [13, 14]. Chen et al. [2, 15] demonstrated in-hand re-orientation for a wide range of objects under palm-up and palm-down orientations of the hand with extrinsic sensing providing dense object feedback. While Khandate et al. [3] showed object-agnostic dexterous finger-gaiting and finger-pivoting skills using precision fingertip grasps and sensing intrinsic to the hand only. Qi et al. [4] used rapid motor adaptation to achieve effective sim-to-real transfer of in-hand manipulation skills for small cylindrical and cube-like objects. Nonetheless, learning these skills still remains sample inefficient.

To improve sample complexity, model-based controllers can be used with model-free reinforcement learning. The natural approach to leverage model-based controllers is to use them as low-level policies in hierarchical reinforcement learning. Li et al. [5] derives model-based controllers for reposing, sliding, and flipping, and then learn a hierarchical policy. However, they demonstrate it only for 2D in-hand manipulation. Veiga et al. [16, 17] used a low-level controller to maintain grasp stability based on tactile feedback but only evaluated for in-grasp manipulation. These approaches all share a common limitation – the low-level controllers are necessary not only during training but also during deployment. This restricts the feedback used by low-level controllers to the sensing modalities available on the hand while simultaneously requiring the controller to be robust to the sim-to-real gap.

Learning a single end-to-end policy while using model-based controllers only during the training phase circumvents this issue. While numerous methods suitable have been proposed, these methods possess limitations for learning dexterous manipulation with sub-optimal controllers.

Learning from demonstrations collected using the model-based controllers can be done either through augmentation [18, 19] or offline RL [20]. However, demonstrations collected using our sub-skill controllers will not be sufficiently exploratory as they can be used for only short rollout horizons.

Imitation learning (IL) methods such as DAgger [21] and AggreVaTe [22] are effective frameworks but only for imitation with a single expert. Moreover, these methods also require the expert to be near-optimal, however, a near-optimal expert controller for dexterous manipulation is both challenging to achieve and expensive to compute.

Adaptions modifying the behavior policy of off-policy RL with sub-skill controllers have also been considered as an alternate approach. Kurenkov et al. [23] proposed AC-Teach framework to sample from multiple experts based on the critic's estimate of the action-value for non-prehensile manipulation. Jeong et al. [24] derived REQ to better utilize the off-policy transitions sampled from a sub-skill expert. Zhang [25] propose value-based policy-composition. Here, we also use the AC-Teach framework, as it provides an effective method to incorporate sub-skill controllers for learning an end-to-end policy, and show that simple controllers and highly sub-optimal controllers can enable exploration even for highly dexterous motor control tasks such as stable in-hand manipulation.

III. Method

We will demonstrate our method for learning dexterous manipulation skills on a task exhibiting a particularly challenging problem in exploration. Specifically, we consider the task of finger-gaiting in-hand manipulation with only fingertip grasps and no support surfaces - a necessary skill for continuous object re-orientation in arbitrary orientations of the hand. Learning this skill purely with reinforcement learning is challenging as random exploration from a fixed state does not sufficiently explore the desired state-space [26]. Due to the complexity and contact-rich nature of the task, it also challenging to use model-based methods to design an expert for finger-gaiting.

Finger-gaiting is a rich skill that consists of a range of behaviors that we can split into two sub-skills: in-grasp manipulation, and contact switching. In-grasp manipulation entails maintaining the object in a stable grasp and reorienting it without breaking or making contact. Therefore, due to limits on the range of motion of the hand, the maximum object rotation is also limited. Contact switching involves

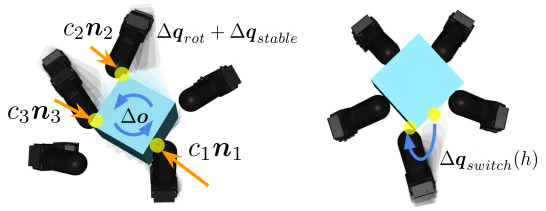


Fig. 2: Sub-skill controllers used for learning finger-gaiting: (left) In-grasp manipulation controller (right) Contact switching controller as described in Sec III-A

.

breaking and making contact to re-grasp the object to further object reorientation via in-grasp manipulation.

Thus, instead of designing a single controller for finger-gaiting, we can decompose it into two controllers, an in-grasp manipulation controller and a contact-switching controller. Still, ensuring optimality and robustness is challenging for these controllers. Nevertheless, sub-optimal model-based controllers obtained from simplified models or heuristics can still generate reasonable actions and can generate exploratory sequences of state transitions.

In this work, we will (i) design simple controllers for in-grasp manipulation and contact switching, and (ii) learn an effective end-to-end policy for finger-gaiting using these sub-optimal controllers by following their actions.

A. Problem definition and sub-optimal controllers

We consider the task of finger-gaiting in-hand manipulation by learning continuous re-orientation about the z-axis in a hand-centric frame. We aim to learn such a policy by rewarding the angular velocity of the object along the z-axis. Let us consider a fully dexterous hand with m fingers and d dofs. Assuming position control, our policy outputs joint target updates from observations $\boldsymbol{s}$:

$$\boldsymbol{s} = [\boldsymbol{q}, \boldsymbol{q}', \boldsymbol{p}_1, .., \boldsymbol{p}_m, \boldsymbol{n}_1, .., \boldsymbol{n}_m, c_1, .., c_m] \tag{1}$$

where $\boldsymbol{q} \in \mathcal{R}^d$, $\boldsymbol{q}' \in \mathcal{R}^d$ are current and target joint positions, $\boldsymbol{p}_i \in \mathcal{R}^3$, $\boldsymbol{n}_i \in \mathcal{R}^3$ are contact positions, contact normal in the global coordinate frame for contacts on all fingertips and $c_i \in \mathcal{R}$ is the magnitude of contact forces. Similar to [3], we train this policy by rewarding the angular velocity of the object about the z-axis when making at least 3 contacts between the object and the fingertips.

As previously discussed, the finger-gaiting skill can be broken down into two sub-skills - in-grasp manipulation and contact switching. We will now design controllers for each.

1) In-grasp manipulation controller (IGM): The in-grasp manipulation sub-skill maintains a stable grasp on the object and, simultaneously, reorients the object about the desired axis. We construct this controller by superposition of two lower-level controllers - one responsible for stability and another responsible for object re-orientation.

To maintain grasp stability, we assume at least three contacts and desire the net object wrench be close to zero. We use the kinematic model of the hand to derive our stability controller as follows. Let S be the set of contact points and $\boldsymbol{n}_1, \ldots, \boldsymbol{n}_k$ ($k \geq 3$) be the contact normals. $\boldsymbol{G}_S$ and $\boldsymbol{J}_S$ are the grasp map matrix and the Jacobian for the set of contacts. Our stability controller keeps the object in equilibrium by minimizing the net wrench applied to the object. This can be achieved by solving the following Quadratic Program (QP):

Algorithm 1 In-grasp manipulation (IGM) controller

1: Get Jacobian, $\boldsymbol{J}_S$ and grasp map matrix $\boldsymbol{G}_S$
2: Compute optimal contact force c_i^*'s for stability from QP
3: Solve for $\Delta\boldsymbol{q}_{stable}$ using Eq 6
4: Solve for $\Delta\boldsymbol{q}_{rot}$ from sampled $\Delta\boldsymbol{o}$ using Eq 7
5: $\psi_{IGM}(\boldsymbol{s}) = \Delta\boldsymbol{q}_{stable} + \Delta\boldsymbol{q}_{rot}$

$$\begin{aligned}
&\text{unknowns: normal force magnitudes } c_i,\ i = 1 \ldots k \\
&\quad \text{minimize } \|\boldsymbol{w}\| \text{ subject to:} \\
&\qquad \boldsymbol{w} = \boldsymbol{G}_S^T \left[c_1\boldsymbol{n_1} \ldots c_k\boldsymbol{n_k}\right]^T && (2) \\
&\qquad c_i \geq 0\ \forall i && (3) \\
&\exists j \text{ such that } c_j = 1 \text{ (ensure non-zero solution)} && (4)
\end{aligned}$$

Let c_i^*'s be the desired contact forces obtained as solution of above QP. We can compute the actions Δq_{stable} needed for stability (expressed as joint position setpoint changes) by solving the following system:

$$\begin{aligned}
\Delta\boldsymbol{p}_i &= \alpha(c_i^* - c_i)\boldsymbol{n}_i && (5) \\
\boldsymbol{J}_S \Delta q_{stable} &= [\Delta\boldsymbol{p}_1, \ldots, \Delta\boldsymbol{p}_k]^T && (6)
\end{aligned}$$

where α is the stability controller gain.

Separately, let $\Delta\boldsymbol{o} \in \mathcal{R}^6$ denote the desired change in object pose. We can compute the action $\Delta\boldsymbol{q}_{rot}$ required to achieve this object re-orientation by solving

$$\boldsymbol{J}_S \Delta\boldsymbol{q}_{rot} = \boldsymbol{G}_S^T \Delta\boldsymbol{o} \tag{7}$$

Finally, $\Delta\boldsymbol{q}_{stable}$ and $\Delta\boldsymbol{q}_{rot}$ are combined together simply via superposition. The complete set of steps involved in the in-grasp manipulation controller is outlined in Alg. 1. When used by itself in a rollout the cumulative reward achieved by the in-grasp manipulation controller before reaching joint limits is far lower than the cumulative reward achievable by a successful finger-gaiting policy within the same duration.

2) Contact switching controller (CS): Our insight here is to simply use a fixed trajectory as a controller,

$$\psi_{CS}(\boldsymbol{s}) = \Delta\boldsymbol{q}_{switch}(h) \tag{8}$$

where $h = 1, \ldots, H$ and H is the length of the controller trajectory.

This trajectory is a simple primitive for breaking and making one contact as described next. We design a trajectory that only breaks and makes contact with one selected finger at a given time even though contact switching as a skill can involve multiple contact switches. We select a finger in contact at random and follow a hand-designed trajectory as shown in Fig 2 for the selected finger to break contact and remake contact with the object at a different location. This trajectory is achieved as shown in Fig 4b. As one can expect, the switching controller by itself collects no meaningful reward.

3) Finger-gaiting controller (FG): Finally, we also construct a third controller. Combining the in-grasp manipulation and contact switching controller we construct a finger-gaiting controller. While a randomly selected fingertip breaks and makes contact, this controller also reorients the object with the fingertips in contact. Although this controller comes closer than the rest, it does not achieve the whole finger-gaiting skill we are looking to learn. Continuous object re-orientation is not possible with this controller as the fingers selected to switch contacts are still selected at random without any coordination.

We construct this controller again by superposition. We superpose in-grasp manipulation and contact switching controller.

$$\psi_{FG}(\boldsymbol{s}) = \psi_{IGM}(\boldsymbol{s}) + \psi_{CS}(\boldsymbol{s}) \tag{9}$$

This finger-gaiting controller also achieves a far lower return relative to a successful finger-gaiting policy.

B. Learning from sub-skill controllers

We use off-policy actor-critic reinforcement learning, where the policy used for episode rollouts (i.e. the behavior policy) can be significantly different from the end-to-end policy being learned (i.e. the learner policy). We use this dichotomy to enable exploration by constructing a behavior policy that is a composition of the learner policy and the sub-skill controllers. Importantly, this approach also sidesteps the need for the controller during deployment - once trained, the learner policy by itself is sufficient for deployment.

We construct the behavior policy to achieve sufficient exploration as follows. Our behavior policy periodically selects between the controllers and learner policy. We bias this selection towards higher value controllers using the critic's action-value estimate of the actions sampled from these controllers. This heuristic allows the probability of selecting a controller action to vary across the state-space. We show that this heuristic of using action-value estimates to select between available controllers and learner policy improves exploration. Optionally, we use value-weighted behavior cloning in addition to the nominal RL objective of the chosen off-policy RL method to update the policy. The method is detailed in the following paragraphs.

Let π_θ represent the learner policy and ψ represent the controller. At every step of the rollout, we query actions from both the learner policy and controllers. Let these actions be $\boldsymbol{a}_\pi$, $\boldsymbol{a}_\psi$. The probability p_ψ of selecting the action from a controller, as well as the probability p_π of selecting the action prescribed by the learner policy, are computed as:

$$p_\psi = \frac{\exp(Q(\boldsymbol{s}, \boldsymbol{a}_\psi))}{\exp(Q(\boldsymbol{s}, \boldsymbol{a}_\pi)) + \exp(Q(\boldsymbol{s}, \boldsymbol{a}_\psi))} \tag{10}$$

$$p_\pi = 1 - p_\psi \tag{11}$$

As $Q(\boldsymbol{a}_\pi, \boldsymbol{s})$ increases, p_ψ decreases, i.e., as the learner policy improves, the probability of following the controller decays exponentially. In practice, we also impose a hard stop, i.e., we stop querying the controllers altogether after a few million steps of training and only use the learner policy for exploration. This point typically corresponds to the convergence of the critic.

Algorithm 2 Learning from sub-optimal controllers

Require: Controller ψ

1: Initialize policy π_θ, Q_ϕ and off-policy RL algorithm with replay buffer $\mathcal{D}$
2: **for** each iteration **do**
3: **for** $t = 0, \ldots, T-1$ **do**
4: **for** every H steps **do**
5: Compute p_π, p_ψ from $\boldsymbol{a}_\pi, \boldsymbol{a}_\psi$ (Eq 10 & 11)
6: Select μ: $\mu \sim \{\pi, \psi\}$ given p_π, p_ψ
7: Follow selected μ for next H steps: $\boldsymbol{a}_t \sim \mu(\boldsymbol{s}_t)$
8: $\mathcal{D} \leftarrow \mathcal{D} \cup \{\boldsymbol{s}_t, \boldsymbol{a}_t, r(\boldsymbol{s}_t, \boldsymbol{a}_t), \boldsymbol{s}_{t+1}\}$
9: **for** each gradient step **do**
10: $\theta \leftarrow \theta - \nabla_\theta \mathcal{L}_{RL}(\theta)$
11: $\theta \leftarrow \theta - \nabla_\theta \mathcal{L}_{BC}(\theta)$ ▷ Optional

Our method is summarized in Alg 2. The first difference w.r.t the standard off-policy learning framework is steps 4-7 when the behavior policy is different from the learning policy and is composed as described previously. The second difference, in step 11, entails updating the learner policy parameters using a value weighted behavior cloning loss $\mathcal{L}_{BC}$ (see Eq 12), in addition to the nominal actor loss $\mathcal{L}_{RL}$ as determined by the off-policy algorithm of choice. We follow a linear decay schedule for β.

$$\mathcal{L}_{BC}(\theta) = \sum_{(\boldsymbol{s},\boldsymbol{a}),\ \boldsymbol{a} \neq \boldsymbol{a}_\pi} \beta \log \pi_\theta(\boldsymbol{a}|\boldsymbol{s}) Q(\boldsymbol{s}, \boldsymbol{a}) \tag{12}$$

IV. Experiments and Results

Our main experimental goals are: (a) illustrating the importance of sampling actions from controllers for enabling exploration and (b) studying the effect of the different sub-optimal controllers proposed.

A. Experimental setup

We use a dexterous hand in a simulated environment as shown in Fig 4a. It is a fully-actuated 15-DOF hand consisting of five fingers where each finger consists of one roll joint followed by two flexion joints. Although our method is effective for a range of objects, in our analysis, we use a cube of side $10cm$ unless otherwise specified.

The key hyperparameters used for the sub-optimal controller are as follows. For the in-grasp manipulation controller, we set the desired change in object pose $\Delta\boldsymbol{o}$ so that it represents a small reorientation about the z-axis. For the contact switching controller, which follows a fixed hand-designed trajectory, the velocity profile of the trajectory is as shown in Fig 4b. It is designed to detach and re-attach the fingertip with the object. The roll-joint and middle-flexion-joint velocities are randomly sampled in the range of $[0.1, 0.4]$ rad/s.

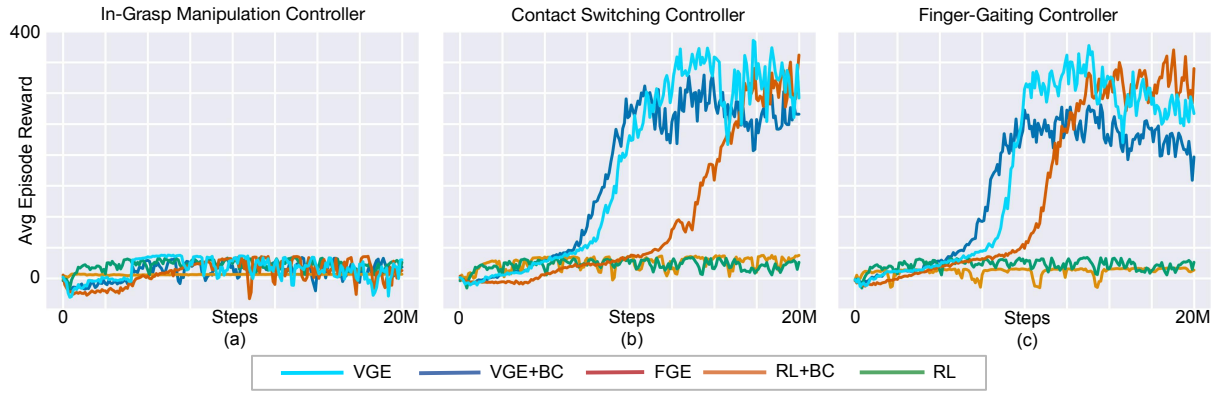
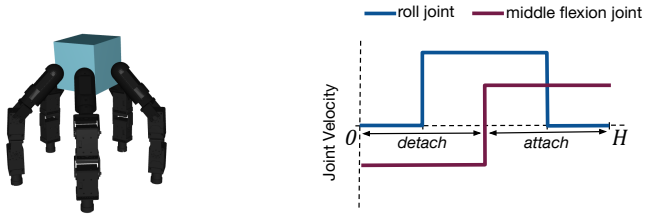


Fig. 3: Training curves for using (a) In-grasp manipulation controller (b) Contact switching controller (c) Finger-gaiting controller over all the evaluation conditions listed in Sec IV-A. Our method that interleaves following controllers with policy (VGE, VGE + BC) learns finger-gaiting, while standard off-policy RL that does not follow controller actions fails. This shows it is essential to follow actions sampled by the sub-skill controllers to enable exploration.



(a) 15-dof hand

(b) Joint velocity for contact switching

Fig. 4: (a) The 5-fingered, 15-dof dexterous hand grasping the cube we use in our experiments (b) Joint velocity profile of the joints of the finger making and breaking contact. It involves detach and attach phases. Note that the distal flexion joint is held fixed.

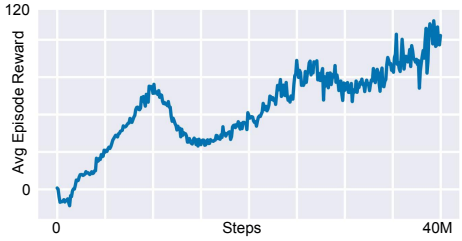


Fig. 5: Training curve for multiple objects using finger-gaiting controller.

TABLE I: Training robustness calculated as the percentage of seeds that successfully learn with about 10 seeds for each method.

	CS Controller	FG Controller
VGE	27%	**100%**
VGE + BC	36%	**100%**
FGE	67%	**73%**

Finally, as our off-policy RL algorithm, we used SAC [27] and TD3 [28] both with similar results. However, for brevity, we show results only for SAC. Note that with SAC, we set the exploration co-efficient $\alpha = 0$ – using any non-zero value fails for all the baselines and also our method(s). Additionally, we do not query the controllers for actions after 4M steps of training in all of our evaluations.

B. Evaluated conditions

We evaluate and compare the following approaches:

1) RL: Standard off-policy RL without the use of controllers and without any modification to the behavior policy or loss function is the first baseline.

2) RL + BC loss: We also test a version of off-policy RL augmented using the same behavior cloning loss w.r.t the behavior policy as used in our method.

3) Fixed Guided Exploration (FGE): We consider this baseline to study the advantage of allowing the probability of following the controllers to vary across the state-space. Unlike our method, controller probability is independent of the state as it is a fixed decay schedule based on training progress.

4) Value Guided Exploration (VGE): This is the method described in this paper, combining sub-skill controllers and value-guided controller sampling.

5) VGE + BC loss: This is a variant of our method that additionally uses behavior cloning with loss as in Eq. (12).

C. Results and discussion

We evaluated the above experimental conditions individually for each of the sub-optimal controllers described in Sec III-A (Fig IVa-c). As we see, our method generally outperforms the baselines considered for each controller as proposed.

The inability of baselines RL + BC loss and RL to learn is important to note. It underscores the key idea of our method that following actions sampled by the sub-skill controllers enables an otherwise very difficult exploration.

Our evaluations also provide insights regarding the properties of the controllers that are critical for exploration. Fig 3 also demonstrates, that contact-switching behavior is critical for such exploration, as this controller enables learning finger-gaiting whenever it is used.

While the in-grasp manipulation (IGM) controller is not critical for learning finger-gaiting, it is still beneficial as it improves training robustness. As seen from Table I, the success rate of learning to gait w.r.t varying seeds is markedly improved which may be attributable to the stability controller used as part of the in-grasp manipulation controller.

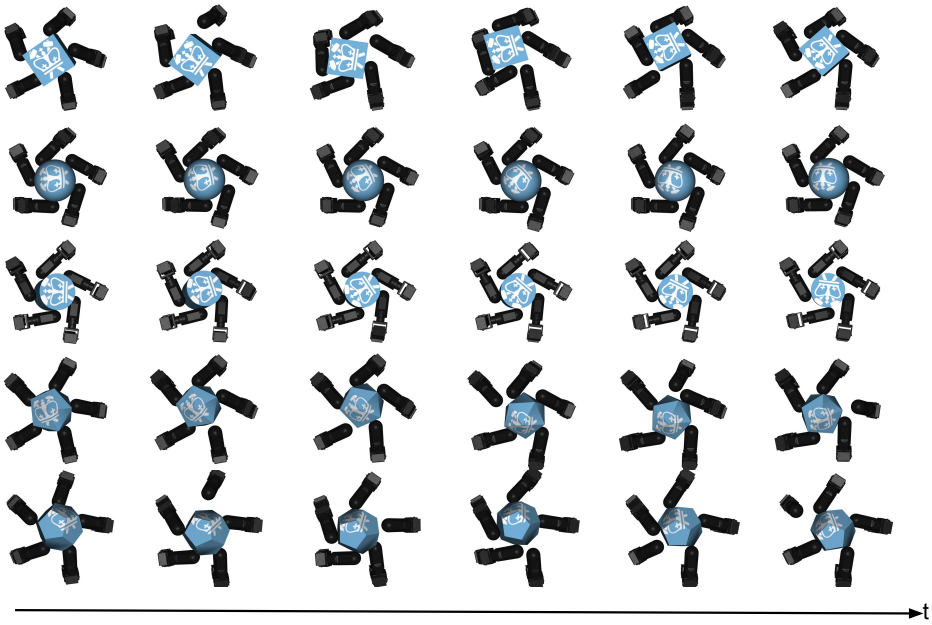


Fig. 6: Keyframes of the gaits achieved for multiple objects simultaneously with a single policy.

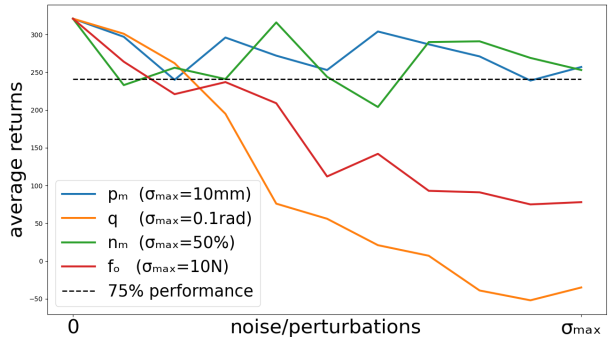


Fig. 7: Average episode returns of a policy trained on the cube with increasing perturbation forces and noise evaluated separately as shown. The policy sustains $0.1rad$ of noise in joint positions $\boldsymbol{q}$, up to 50% error in contact normal $\boldsymbol{n}_m$ and a large error of $3mm$ in contact positions $\boldsymbol{p}_m$ – all while suffering only 25% percent drop in average episode returns.

Finally, we can also learn a single finger-gaiting policy for multiple objects with our method. In Fig 5, we show that the finger-gaiting (FG) controller enables learning the finger-gaiting policy simultaneously for the sphere, cylinder, dodecahedron, icosahedron, and cube. The keyframes of the gaits achieved with this policy as visualized in Fig 6.

D. Sim-to-real transfer feasibility

While we do not attempt sim-to-real transfer in this work, we believe that these policies can be transferred to the real hand using established methods from literature. As demonstrated by recent work [1, 4, 29, 30], domain randomization has been effective for transferring dexterous manipulation skills learned in simulation to the real hand. To investigate if our policies are suitable for such methods, we also tested their robustness to both perturbation forces on the object and noise in sensory feedback. As shown in Fig 7, our policies can sustain large perturbation forces equivalent to the full weight of the object, and are also robust to a high degree of sensor noise. This characteristic leads us to believe that training with a curriculum of domain randomization will make transferring these robust policies to the real hand viable, and we hope to demonstrate this in future work.

V. Conclusion

Our work proposed the use of model-based controllers to assist exploration in reinforcement learning of dexterous manipulation tasks. To this end, we use off-policy RL and take advantage of the freedom with the construction of behavior policy to follow actions from the controller while collecting rollout trajectories to update the learning policy with such data to learn a successful policy.

We evaluated our method with the challenging dexterous manipulation task of learning finger-gaiting in-hand manipulation with only fingertip grasps. We designed simple controllers for key sub-skills in finger-gaiting - in-grasp manipulation and contact switching - and used these to enable learning effective finger-gaiting skills without any other additional means of improving exploration. Moreover, we show that our method benefits from model-based controllers even if the controllers are significantly sub-optimal.

More generally, we demonstrated a method to use domain expertise for learning dexterous manipulation tasks with improved sample efficiency. We believe our method of using sub-skill controllers for exploration is a promising approach for achieving dexterous skills with complex dynamics. While this work uses only model-based controllers, our method can also use learned sub-skill experts when available. It can also be extended to consider multiple experts. We hope to explore these promising directions in future work.

REFERENCES

[1] OpenAI, Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Jozefowicz, Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, Jonas Schneider, Szymon Sidor, Josh Tobin, Peter Welinder, Lilian Weng, and Wojciech Zaremba. "Learning Dexterous In-Hand Manipulation". In: (Aug. 2018). arXiv: 1808.00177 [cs.LG].

[2] Tao Chen, Megha Tippur, Siyang Wu, Vikash Kumar, Edward Adelson, and Pulkit Agrawal. "Visual Dexterity: In-hand Dexterous Manipulation from Depth". In: (Nov. 2022). arXiv: 2211.11744 [cs.RO].

[3] Gagan Khandate, Maximilian Haas-Heger, and Matei Ciocarlie. "On the Feasibility of Learning Finger-gaiting In-hand Manipulation with Intrinsic Sensing". In: *2022 International Conference on Robotics and Automation (ICRA)*. May 2022, pp. 2752–2758.

[4] Haozhi Qi, Ashish Kumar, Roberto Calandra, Yi Ma, and Jitendra Malik. "In-Hand Object Rotation via Rapid Motor Adaptation". In: (Oct. 2022). arXiv: 2210.04887 [cs.RO].

[5] Tingguang Li, Krishnan Srinivasan, Max Qing-Hu Meng, Wenzhen Yuan, and Jeannette Bohg. "Learning Hierarchical Control for Robust In-Hand Manipulation". In: (Oct. 2019). arXiv: 1910.10985 [cs.RO].

[6] Susanna Leveroni and Kenneth Salisbury. "Reorienting Objects with a Robot Hand Using Grasp Gaits". In: *Robotics Research*. Springer London, 1996, pp. 39–51.

[7] L Han and J C Trinkle. "Dextrous manipulation by rolling and finger gaiting". In: *Proceedings. 1998 IEEE International Conference on Robotics and Automation (Cat. No.98CH36146)*. Vol. 1. May 1998, 730–735 vol.1.

[8] Jean-Philippe Saut, Anis Sahbani, Sahar El-Khoury, and Veronique Perdereau. "Dexterous manipulation planning using probabilistic roadmaps in continuous grasp subspaces". In: *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*. Oct. 2007, pp. 2907–2912.

[9] T Omata and M A Farooqi. "Regrasps by a multifingered hand based on primitives". In: *Proceedings of IEEE International Conference on Robotics and Automation*. Vol. 3. Apr. 1996, 2774–2780 vol.3.

[10] Yongxiang Fan, Te Tang, Hsien-Chung Lin, Yu Zhao, and Masayoshi Tomizuka. "Real-Time Robust Finger Gaits Planning under Object Shape and Dynamics Uncertainties". In: (Oct. 2017). arXiv: 1710.10350 [cs.RO].

[11] Balakumar Sundaralingam and Tucker Hermans. "Geometric In-Hand Regrasp Planning: Alternating Optimization of Finger Gaits and In-Grasp Manipulation". In: (Apr. 2018). arXiv: 1804.04292 [cs.RO].

[12] Claire Chen, Preston Culbertson, Marion Lepert, Mac Schwager, and Jeannette Bohg. "TrajectoTree: Trajectory Optimization Meets Tree Search for Planning Multi-contact Dexterous Manipulation". In: (Sept. 2021). arXiv: 2109.14088 [cs.RO].

[13] Viktor Makoviychuk, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin, David Hoeller, Nikita Rudin, Arthur Allshire, Ankur Handa, and Gavriel State. "Isaac Gym: High Performance GPU-Based Physics Simulation For Robot Learning". In: (Aug. 2021). arXiv: 2108.10470 [cs.RO].

[14] Arthur Allshire, Mayank Mittal, Varun Lodaya, Viktor Makoviychuk, Denys Makoviichuk, Felix Widmaier, Manuel Wüthrich, Stefan Bauer, Ankur Handa, and Animesh Garg. "Transferring Dexterous Manipulation from GPU Simulation to a Remote Real-World TriFinger". In: (Aug. 2021). arXiv: 2108.09779 [cs.RO].

[15] Tao Chen, Jie Xu, and Pulkit Agrawal. "A System for General In-Hand Object Re-Orientation". Nov. 2021.

[16] Filipe Veiga, Benoni B Edin, and Jan Peters. "In-Hand Object Stabilization by Independent Finger Control". In: (June 2018). arXiv: 1806.05031 [cs.RO].

[17] Filipe Veiga, Riad Akrour, and Jan Peters. "Hierarchical Tactile-Based Control Decomposition of Dexterous In-Hand Manipulation Tasks". en. In: *Front Robot AI* 7 (Nov. 2020), p. 521448.

[18] Aravind Rajeswaran, Vikash Kumar, Abhishek Gupta, Giulia Vezzani, John Schulman, Emanuel Todorov, and Sergey Levine. "Learning complex dexterous manipulation with deep reinforcement learning and demonstrations". In: (Sept. 2017). arXiv: 1709.10087 [cs.LG].

[19] Ilija Radosavovic, Xiaolong Wang, Lerrel Pinto, and Jitendra Malik. "State-Only Imitation Learning for Dexterous Manipulation". In: *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Sept. 2021, pp. 7865–7871.

[20] Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. "Offline Reinforcement Learning: Tutorial, Review, and Perspectives on Open Problems". In: (May 2020). arXiv: 2005.01643 [cs.LG].

[21] Stephane Ross, Geoffrey J Gordon, and J Andrew Bagnell. "A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning". In: (Nov. 2010). arXiv: 1011.0686 [cs.LG].

[22] Stephane Ross and J Andrew Bagnell. "Reinforcement and Imitation Learning via Interactive No-Regret Learning". In: (June 2014). arXiv: 1406.5979 [cs.LG].

[23] Andrey Kurenkov, Ajay Mandlekar, Roberto Martin-Martin, Silvio Savarese, and Animesh Garg. "AC-Teach: A Bayesian Actor-Critic Method for Policy Learning with an Ensemble of Suboptimal Teachers". In: (Sept. 2019). arXiv: 1909.04121 [cs.LG].

[24] Rae Jeong, Jost Tobias Springenberg, Jackie Kay, Daniel Zheng, Yuxiang Zhou, Alexandre Galashov, Nicolas Heess, and Francesco Nori. "Learning Dexterous Manipulation from Suboptimal Experts". In: (Oct. 2020). arXiv: 2010.08587 [cs.RO].

[25] Haichao Zhang. *PEX: Policy Expansion for Bridging Offline-to-Online Reinforcement Learning (ICLR23)*. en.

[26] Gagan Khandate, Maxmillian Haas-Heger, and Matei Ciocarlie. "On the Feasibility of Learning Finger-gaiting In-hand Manipulation with Intrinsic Sensing". In: (Sept. 2021). arXiv: 2109.12720 [cs.RO].

[27] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. "Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor". In: (Jan. 2018). arXiv: 1801.01290 [cs.LG].

[28] Scott Fujimoto, Herke van Hoof, and David Meger. "Addressing Function Approximation Error in Actor-Critic Methods". In: (Feb. 2018). arXiv: 1802.09477 [cs.AI].

[29] Tao Chen, Megha Tippur, Siyang Wu, Vikash Kumar, Edward Adelson, and Pulkit Agrawal. "Visual Dexterity: In-hand Dexterous Manipulation from Depth". In: (Nov. 2022). arXiv: 2211.11744 [cs.RO].

[30] Gagan Khandate, Siqi Shang, Eric T Chang, Tristan L Saidi, Johnson Adams, and Matei Ciocarlie. "Sampling-based Exploration for Reinforcement Learning of Dexterous Manipulation". In: *Robotics: Science and Systems XIX*. Vol. 19. July 2023.