Dexterous Manipulation with Deep Reinforcement Learning: Efficient, General, and Low-Cost

Henry Zhu*[1] Abhishek Gupta*[1] Aravind Rajeswaran[2] Sergey Levine[1] Vikash Kumar[3]
[1] UC Berkeley [2] University of Washington [3] Google Brain

***Abstract*— Dexterous multi-fingered robotic hands can perform a wide range of manipulation skills, making them an appealing component for general-purpose robotic manipulators. However, such hands pose a major challenge for autonomous control, due to the high dimensionality of their configuration space and complex intermittent contact interactions. In this work, we propose deep reinforcement learning (deep RL) as a scalable solution for learning complex, contact rich behaviors with multi-fingered hands. Deep RL provides an end-to-end approach to directly map sensor readings to actions, without the need for task specific models or policy classes. We show that contact-rich manipulation behavior with multi-fingered hands can be learned by directly training with model-free deep RL algorithms in the real world, with minimal additional assumption and without the aid of simulation. We learn a variety of complex behaviors on two different low-cost hardware platforms. We show that each task can be learned entirely from scratch, and further study how the learning process can be further accelerated by using a small number of human demonstrations to bootstrap learning. Our experiments demonstrate that complex multi-fingered manipulation skills can be learned in the real world in about 4-7 hours for most tasks, and that demonstrations can decrease this to 2-3 hours, indicating that direct deep RL training in the real world is a viable and practical alternative to simulation and model-based control.** `https://sites.google.com/view/deeprl-handmanipulation`

I. Introduction

A long standing goal in robotics is to create general purpose robotic systems that operate in a wide variety of human-centric environments such as homes and hospitals. For robotic agents to be competent in such unstructured environments, versatile manipulators like multi-fingered hands are needed in order to cope with the diversity of tasks presented by human-centric settings. However, the versatility of multi-fingered robotic hands comes at the price of high dimensional configuration spaces and complex finger-object contact interactions, which makes modeling and controller synthesis particularly challenging.

Reinforcement learning (RL) and optimal control techniques provide generic paradigms for optimizing complex controllers that are hard to specify by hand. In particular, model-free RL provides a way to optimize controllers end-to-end without any explicit modeling or system identification. This reduces human engineering effort and produces controllers that are directly adapted to the physical environment, which makes for a scalable and general approach to robotic manipulation: a large repertoire of skills can be acquired by directly learning in different situations in the real world with only the task reward specified by the user. Such systems do not require an explicit model of the robot, calibration of the sensors, or other manual characterization that is typically needed for model-based methods.

Fig. 1: We demonstrate that DRL can learn a wide range of dexterous manipulation skills with multi-fingered hands, such as opening door with flexible handle, rotating a cross-shaped valve, and rotating the same valve but with a deformable foam handle, which presents an additional physical challenge, and box flipping.

In this work, we study how model-free RL can be scaled up to learn a variety of manipulation behaviors with multi-fingered hands directly in the real world, using general-purpose neural network policy representations and without manual controller or policy class design. We conduct our experiments with low cost multi-fingered manipulators, and show results on tasks such as rotating a valve, flipping a block vertically, and door opening. Somewhat surprisingly, we find that successful neural network controllers for each of these tasks can be trained directly in the real world in about 4-7 hours for most tasks.

We also show that we can further accelerate the learning process by incorporating a small number of human demonstrations, building on the recently proposed DAPG [1] algorithm. Using 20 demonstrations obtained through kinesthetic teaching, the learning time can be brought down to around $2-3$ hours which corresponds to a 2x speedup. Together, these

*The first two authors contributed equally to this work.

results establish that model-free deep RL and demonstration-driven acceleration provide a viable approach for real-world learning of diverse manipulation skills.

The main contribution of this work is to demonstrate that model-free deep reinforcement learning can learn contact rich manipulation behaviors directly for low-cost multi-fingered hands directly in the real world. We demonstrate this on two different low cost robotic hands, manipulating both rigid and deformable objects, and performing three distinct tasks. We further demonstrate that a small number of demonstrations can accelerate learning, and analyze a number of design choices and performance parameters.

II. Related Work

In recent years, multi-fingered robotic hands have become more and more widespread as robotic manipulation has progressed towards more challenging tasks. A significant portion of this research has focused on the physical design of anthropomorphic hands, and designing simple controllers for these hands. The majority of these manipulators [2], [3] are custom, expensive, heavily instrumented, and fragile. In contrast, the results presented in this work are on low-cost (and relatively more robust) commodity hardware, with limited sensing and actuation capabilities. As our method trains directly in the real world, the resulting solution encapsulates the sensory and control inaccuracies that are associated with such hardware thereby making them more effective and deployable.

Much of the work on robotic hands has focused on grasping [4], as compared to other dexterous skills [5]. Such work is often focused on achieving stable grasps by explicitly reasoning about the stability of the grasp using geometric and analytic metrics [6], [7], [8]. While considerable progress has been made on the theoretical side [9], sensing and estimation challenges (visual occlusions and limited tactile sensing) has limited their applicability in the real world considerably. In this work, we consider tasks which require significant finger coordination and dexterity, making the manual design of controllers very challenging. Our proposed solution using model-free deep RL alleviates the need for manual controller design or modeling.

In this work, we study the use of model-free deep reinforcement learning algorithms to learn manipulation policies for dexterous multi-finger hands. In simulation, [10], [11], [12] have shown the ability to synthesize complex behaviors when provided with a ground truth model. This model is rarely available in the real world. Techniques such as [13], [14] get around this problem by learning locally-linear models and deploying model-based RL method to learn instance specific linear controllers demonstrating turning rods and arranging with beads. [15] show the ability to learn in-hand repositioning in the real world, with a reinforcement learning method (NPREPS) and non-parametric control policies. Similar methods have also been used for whole-arm manipulation [16]. While these methods train in the real world, the resulting policies involve relatively simple individual motions. In contrast to these model-free experiments, our results show that deep RL can learn complex contact-rich behaviors with finger gaits, including with combined hand and arm control. In contrast to the model-based methods, we show that this can be done directly in the real world without any model.

We also show how the learning process can be accelerated using a small number of human demonstrations. Accelerating reinforcement learning with demonstrations has been the subject of a number of works in the reinforcement learning community [17], [16], [18], [19], [1], [20], [21] but these methods have not been applied to real world manipulation with multi-fingered hands. We show that this is indeed possible and very effective using algorithms which combine behavior cloning and policy gradient. We build on the algorithm proposed by us in prior work [1], and show that this approach can indeed scale to real world dexterous manipulation, with significant acceleration benefits.

Another related line of research seeks to transfer policies trained in simulation into the real world. Aside from rigorous system identification, a recent class of methods has focused on randomizing the simulation, both for physical [22] and visual [23] transfer. This approach has been employed for manipulation [24], visual servoing [25], locomotion [26], and navigation [23]. Recent concurrent work has also applied this approach to multi-fingered hands for an object rotation task [27] with about 50 hours of computation, equivalent to 100 years of experience. In contrast, our approach trains directly in the real world in a few hours without the need for manual modeling. We discuss the relative merits of real-world and simulated training in Section VI-D.

III. Hardware Setup

In order to demonstrate the generalizable nature of the model-free deep RL algorithms, we consider two different hardware platforms: a custom built 3 fingered hand, referred to as the Dynamixel claw (Dclaw), and a 4 fingered Allegro hand. Both hands are relatively cheap, especially the Dclaw, which costs under $\$2,500$ to build. For several experiments, we also mounted the hands on a Sawyer robot arm to allow for a larger workspace.

a) Dynamixel Claw: The Dynamixel claw (Dclaw) is custom built using Dynamixel servo motors. It is a powerful, low latency, position controlled 9 DoF manipulator which costs under $\$2,500$ to construct. Dclaw is robust and is able to run up to 24 hours without intervention or hardware damage.

b) Allegro Hand: The Allegro hand is a 4 fingered anthropomorphic hand, with 16 degrees of freedom, and can handle payloads of up to 5 kg. This hand uses DC motors for actuation and can be either torque or position controlled using a low level PID. The Allegro hand costs on the order of $\$15,000$.

IV. Tasks

While the approach we describe for learning dexterous manipulation is general and broadly applicable, we consider three distinct tasks in our experimental evaluation - valve rotation, box flipping, and door opening. These tasks involve

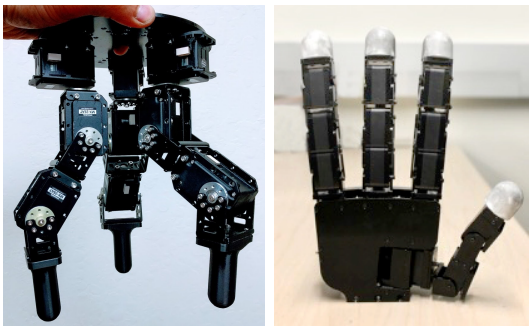

Fig. 2: Left: 3 finger Dynamixel claw. Right: 4 finger anthropomorphic Allegro hand

challenging contact patterns and coordination, and are inspired by everyday hand manipulations. We approach these problems using reinforcement learning, modeling them as Markov decision processes (MDPs), which provide a generic mathematical abstraction to model sequential decision making problems. The goal in reinforcement learning is to learn a control policy which maximizes a user-provided reward function. This reward function is defined independently for each of our tasks as described below.

a) Valve Rotation: This task involves turning a valve or faucet to a target position. The fingers must cooperatively push and move out of the way, posing an exploration challenge. Furthermore the contact forces with the valve complicate the dynamics. For our task, the valve must be rotated from 0° to 180°.

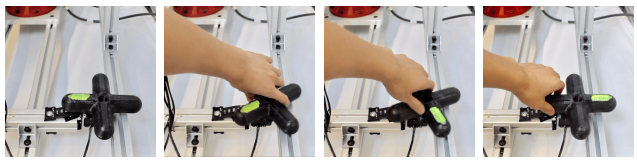

Fig. 3: Illustration of valve rotation

The state space consists of all the joint angles of the hand, the current angle of rotation of the valve $[\theta_{\text{valve}}]$, the distance to the goal angle $[d\theta]$, and the last action taken. The action space is joint angles of the hand and the reward function is

$$r = -|d\theta| + 10 * \mathbb{1}_{\{|d\theta|<0.1\}} + 50 * \mathbb{1}_{\{|d\theta|<0.05\}}$$
$$d\theta := \theta_{\text{valve}} - \theta_{\text{goal}}$$

We define a trajectory as a success if $|d\theta| < 20^\circ$ for at least 20% of the trajectory.

b) Vertical box flipping: This task involves rotating a rectangular box, which freely spins about its long axis, from 0° to 180°. This task also involves learning alternating coordinated motions of the fingers such that while the top finger is pushing, the bottom two move out of the way, and vice versa.

The state space consists of all the joint angles of the hand, the current angle of rotation of the box $[\theta_{\text{box}}]$, the distance in angle to the goal, and the last action taken. The action space consists of the joint angles of the hand and the reward function is

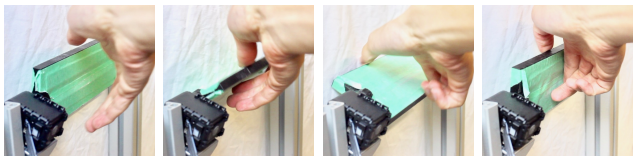

Fig. 4: Illustration of box flipping

$$r = -|d\theta| + 10 * \mathbb{1}_{\{|d\theta|<0.1\}} + 50 * \mathbb{1}_{\{|d\theta|<0.05\}}$$
$$d\theta := \theta_{\text{box}} - \theta_{\text{goal}}$$

We define a trajectory as a success if $|d\theta| < 20^\circ$ for at least 20% of the trajectory.

c) Door opening: This task involves both the arm and the hand working in tandem to open a door. The robot must learn to approach the door, grip the handle, and then pull backwards. This task has more degrees of freedom given the additional arm, and involves the sequence of actions: going to the door, gripping the door, and then pulling away.

Fig. 5: Opening door with flexible handle

The state space is all the joint angles of the hand, the Cartesian position of the arm, the current angle of the door, and last action taken. The action space is the position space of the hand and horizontal position of the wrist of the arm. The reward function is provided as

$$r = -(d\theta)^2 - (x_{\text{arm}} - x_{\text{door}})$$
$$d\theta := \theta_{\text{door}} - \theta_{\text{closed}}$$

We define a trajectory as a success if at any point $d\theta > 30^\circ$.

A. Dynamixel Driven State Estimation

For these tasks, we solve both the problem of state estimation and resetting using a setup with objects augmented with Dynamixel servo motors. These motors serve the dual purpose of resetting objects (such as the valve, box, or door) to their original positions and measuring the state of the object (angle of rotation or position).

Fig. 6: Dynamixel driven sensing and reset mechanisms. Left to Right: rigid valve, box, door.

V. Algorithms

In this work, we show that the tasks described in the previous section can be solved using model-free on-policy reinforcement learning algorithms. This requires 7 hours for valve turning, 4 for box flipping and 16 hours for door opening. We also demonstrate that the learning time can be significantly reduced by using a small number of human demonstrations.

A. Policy Gradient

Reinforcement learning algorithms operate within the framework of Markov decision processes. An MDP is described using the tuple: $\mathcal{M} = \{\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{T}, \rho_0, \gamma\}$. Here, $\mathcal{S} \subseteq \mathbb{R}^n$ and $\mathcal{A} \subseteq \mathbb{R}^m$ represent the state and action spaces respectively. $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function. $\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}_{\geq 0}$ represents the stochastic unknown transition dynamics. The goal in reinforcement learning is to find a policy π that describes how to choose actions in any given state, such that we maximize the sum of expected rewards. The performance of a policy is given by:

$$\eta(\pi) = \mathbb{E}_{\tau \sim \mathcal{M}^\pi} \left[\sum_{t=0}^{\infty} \gamma^t r_t \right]. \tag{1}$$

In this work, we parameterize the policy using a neural network, and use a gradient ascent based approach to optimize (1). Simply performing gradient ascent on (1) is often referred to as vanilla policy gradient (REINFORCE) [28], given by:

$$\nabla_\theta \eta = \mathbb{E}_{\mathcal{M}^{\pi_\theta}} \left[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t) \sum_{t'=t}^{T} r(s_t, a_t) \right] \tag{2}$$

REINFORCE is known to be slow and ineffective. Natural policy gradient methods capture curvature information about the optimization landscape, thereby stabilizing the optimization process and enabling faster convergence. The natural policy gradient is computed by preconditioning the REINFORCE gradient with the inverse of the Fisher Information Matrix, which is defined as

$$\mathcal{F}(\theta) = \mathbb{E}_{s,a}[\nabla_\theta \log \pi_\theta(a|s) \nabla_\theta \log \pi_\theta(a|s)^T]. \tag{3}$$

Thus, the natural policy gradient update rule is given by:

$$\theta_{i+1} = \theta_i + \alpha \mathcal{F}^{-1}(\theta_i) \nabla_{\theta_i} \eta \tag{4}$$

There are numerous variants of the natural policy gradient method [29], [30]. For simplicity, we use the truncated natural policy gradient (TNPG) as described in [31].

B. Demonstration Augmented Policy Gradient

While the NPG algorithm is guaranteed to converge asymptotically to at least a locally optimal policy, the rate of convergence could be very slow. In particular, each update step in NPG requires interacting with the environment to collect on-policy data which may be slow for some tasks. In such cases, we would like to accelerate the learning process by using various forms of prior knowledge.

One way to incorporate human priors is through the use of demonstration data, obtained through say kinesthetic teaching. In prior work, we developed the demonstration augmented policy gradient (DAPG) algorithm which combines reinforcement learning with imitation learning using the demonstration data. Let $\mathcal{D} = \{(s_t^i, a_t^i, r_t^i)\}$ denote the demonstration dataset. Let $\mathcal{D}^\pi$ denote the on-policy dataset obtained by rolling out π. DAPG starts by pre-training the policy π with behavior cloning on this demonstration data $\mathcal{D}$. This pre-trained policy π is subsequently finetuned using an augmented policy gradient.

DAPG first constructs an augmented "vanilla" gradient as:

$$g_{aug} = \sum_{(s,a) \in \mathcal{D}^\pi} \nabla_\theta \ln \pi_\theta(a|s) A^\pi(s,a) + \sum_{(s,a) \in \mathcal{D}} \nabla_\theta \ln \pi_\theta(a|s) w(s,a). \tag{5}$$

We choose $w(s,a) = \lambda_0 \lambda_1^k \max_{(s',a') \in \mathcal{D}^\pi} A^\pi(s',a')$, where λ_0 and λ_1 are hyperparameters, and k is the iteration counter. Subsequently, the policy is updated by preconditioning this augmented gradient with the Fisher information matrix as described in Section V-A

The second term in g_{aug} encourages the policy to be close to the actions taken by experts on states visited by the experts, throughout the learning process. Thus, it can be interpreted as reward shaping with a shaping similar to a trajectory tracking cost. This DAPG algorithm has been shown to significantly accelerate the learning by improving exploration, and exceed the performance of the demonstrations while still retaining their stylistic aspects – all in simulation. In this work, we demonstrate that this algorithm provides a practical and useful way of accelerating deep RL on real hardware to solve challenging manipulation problems.

VI. Experimental Results and Analysis

The goal of our experiments is to empirically address the following research questions:

- Can model-free deep RL provide a practical means for learning various dexterous manipulation behaviors directly in the real world?
- Can model-free deep RL algorithms learn on different hardware platforms and on different physical setups?
- Can we accelerate the learning process using a small number of demonstrations obtained through kinesthetic teaching?
- How do particular design choices of the reward function and actuation space affect learning?

To do so, we utilize the tasks in Section IV and algorithms described in Section V. Additional details can be found at the supplementary website `https://sites.google.com/view/deeprl-handmanipulation`

A. Model-Free Deep RL

First, we explore the performance of model-free deep reinforcement learning on our suite of hardware tasks. The learning progress is depicted in Fig 10 and also in the accompanying video. We find that, somewhat surprisingly, model-free deep RL can acquire coherent manipulation skills

in the time scales of a few hours (7 hours for turning a valve, 4 hours to flip a box, 16 hours for opening a door). These training times are evaluated once the deterministic policy achieves 100% success rate over 10 evaluation rollouts, according to the success metrics defined in Section IV (Fig 16). The algorithm is robust and did not require extensive hyperparameter optimization. The only hyperparameter that was tuned was the initial variance of the policy for exploration. We analyze specific design choices in the reward function and actuation scheme in Section VI-E.

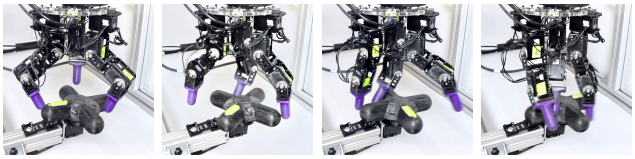

Fig. 7: Visualization of Dclaw policy that has learned to turn a valve after 7 hours of training. The fingers learn to alternately move in and out to turn the valve.

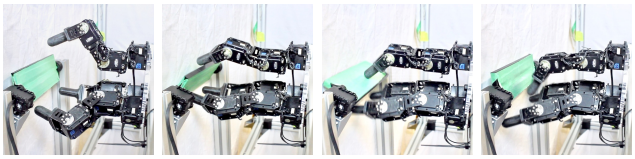

Fig. 8: Visualization of Dclaw that has learned to flip a box after 4 hours of training. The Dclaw learns to extend its bottom two fingers and push its top finger forwards, then lower its bottom two fingers while pushing downwards with its top finger.

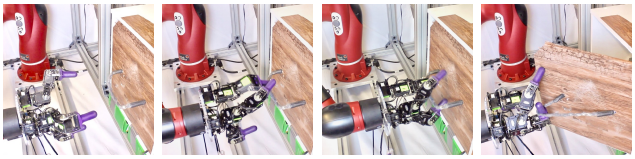

Fig. 9: Visualization of a policy that has learned to open a door after 16 hours of training. The robot learns to move towards the door, grasp the deformable handle, and then pull the door open.

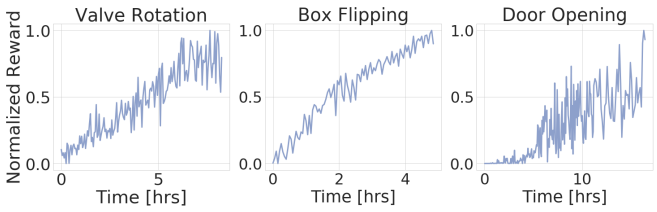

Fig. 10: Learning progress with model-free RL from scratch using the NPG algorithm. The policies reach 100% average success rates after 7.4 hours for valve rotation and 4.1 hours for box flipping and 15.6 hours for door opening.

We find that for the valve and the box flipping, the learning is able to monotonically improve on the continuous reward signal, whereas for the door the learning is more challenging, given that reward is only obtained when the door is actually opened. The agent has to consistently pull open the door in order to see the reward, leading to the large spikes in learning as seen in Fig. 10.

The learned finger gaits that we observe do have interesting coordination patterns. The tasks require that the fingers move quickly and in a coordinated way so as to rotate the objects and then move out of the way. This behavior can be appreciated in the accompanying video on the supplementary website.

B. Learning on Different Hardware and Different Materials

To illustrate the ability of model-free RL algorithms to be applied easily to new scenarios and robots, we evaluated the valve task using the exact same deep RL algorithm with a different hand – the 4-fingered Allegro hand. This hand has 16 DoFs and is also anthropomorphic. We find that the Allegro hand was able to learn this task, as illustrated in Fig. 13, in comparable time as the Dclaw.

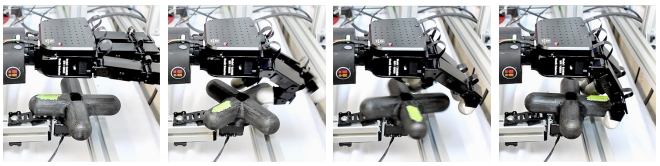

Fig. 11: The Allegro hand learns to rotate a valve 180°.

Both systems are able to quickly learn the right behavior with the same hyperparameters. While morphology does indeed have an effect on rate of learning, we find that it is not a hindrance to eventually learning the task. The easy adaptability of these algorithms is extremely important, since we don't need to construct an accurate simulation or model of each new robot, and can simply run the same algorithm repeatedly to learn new behaviors.

Besides changing the robot's morphology, we can also modify the object that is manipulated. We evaluate whether model-free RL algorithms can be effective at learning with a different valve material, such as soft foam (Fig. 12). The contact dynamics with such a deformable material are hard to simulate and the hand can deform the valve in many different directions, making the actual manipulation task challenging. We see that model-free RL is able to learn manipulation with the foam valve effectively, even generating behaviors that exploit the deformable structure of the object.

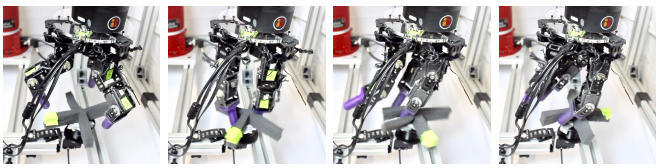

Fig. 12: DClaw learns to rotate a foam valve despite its deformable structure. The claw learns to focus its manipulation on the center of the valve where there is more rigidity.

C. Accelerating Learning with Demonstrations

While we find that model-free deep RL is generally practical in the real world, the number of samples can be

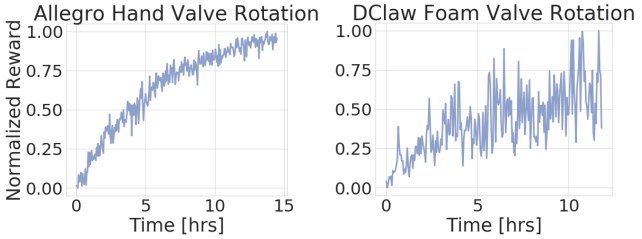

Fig. 13: Left: learning progress of Allegro hand rotating rigid screw. Right: learning progress of Dclaw rotating the foam valve. The rewards have been normalized such that a random policy achieves score of 0 and the final trained policy achieves a score of 1.

further reduced by employing demonstrations, as discussed in Section V-B. In order to understand the role of demonstrations, we collected 20 demonstrations for each task via kinesthetic teaching.

Fig. 14: Kinesthetic teaching was used to obtain demonstrations with the Dclaw.

These demonstrations are slow and suboptimal, but can still provide guidance for exploration and help guide the learning process. With only a few demonstrations, we see that demonstration augmented policy gradient (DAPG) can speed up learning significantly, dropping learning time by 2x (Fig 15). We record training times across tasks using the previously defined success metrics in (Fig 16).

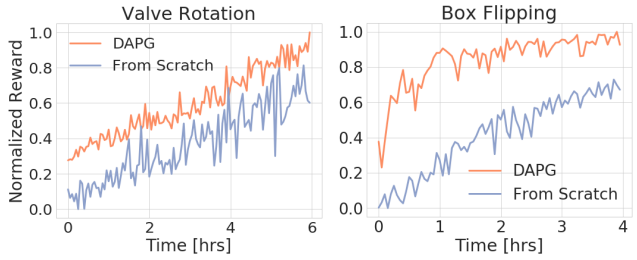

Fig. 15: Learning progress with training from scratch using NPG and using DAPG. The performances have been normalized such that a random policy achieves a score of 0 and the best DAPG policy gets a score of 1.

The efficiency of DAPG comes from the fact that the behavior cloning initialization gives the agent a rough idea of how to solve the task, and the augmented loss function guides learning through several iterations. The behaviors learned are also more gentle and legible to humans than behaviors learned via training from scratch, which is clearly displayed in the accompanying video.

Fig. 16: Training Times Across Tasks [hrs]. Training time is determined using the success metrics defined in Section IV. A training run is complete once the deterministic policy achieves 100% success rate over 10 evaluation rollouts.

Task	From Scratch	DAPG
Valve	7.4	3.0
Box	4.1	1.5
Door	15.6	—

To better understand the learned behaviors, we also evaluated the robustness of these behaviors to variations in the initial position of the valve, and to noise injected into actions and observations (Fig 17).

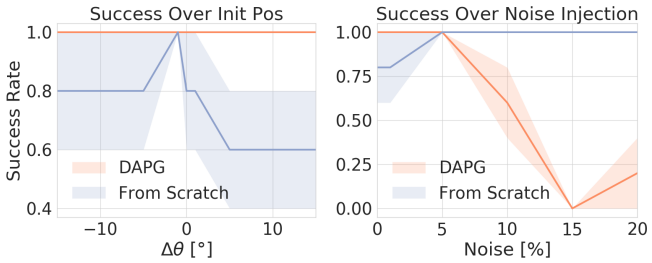

Fig. 17: Plots showing robustness of DAPG vs learning from scratch for the valve rotation task. Left: Variation of success with change in valve initial position (degrees). Right: Variation of success with x% uniformly random noise injected into the observation and action space. DAPG is more robust with change in initial valve position, but is less robust as we add more noise.

We also considered collecting demonstrations from a wider range of initial configurations of the environment, and using this wider demo set for DAPG. The demonstrations were collected with initial valve positions in the range $[-\frac{\pi}{4}, \frac{\pi}{4}]$. This paradigm also works well (Fig 18). Unsurprisingly, it is not as effective as using a number of demonstrations in the same environment configuration, but is still able to learn well. (Fig 15).

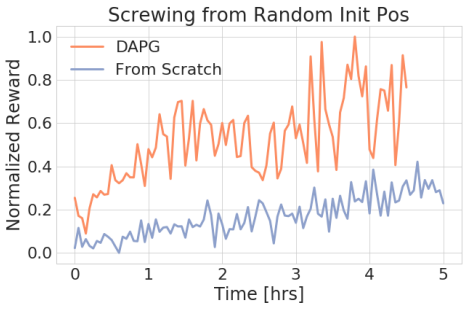

Fig. 18: Training DClaw to turn a valve from a single randomly sampled initial position between [-45°, 45°] to 180°. DAPG was trained with 20 demos that turned the valve from initial positions sampled uniformly at random between [-45°, 45°] to 180°.

D. Performance with Simulated Training

While the main goal of this work is to study how model-free RL can be used to learn complex policies directly on

real hardware, we also evaluated training in simulation and transfer, employing randomization to allow for transfer [22], [23]. This requires modeling the task in a simulator and manually choosing the parameters to randomize.

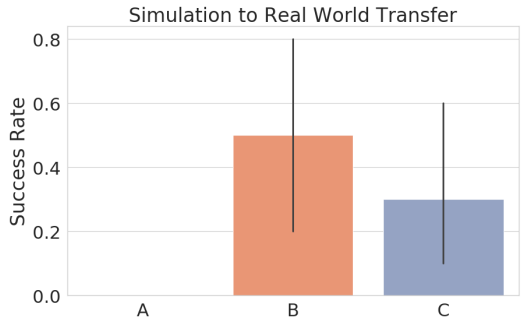

Fig. 19: Success rates using different sim2real transfer strategies for valve turning with Dclaw. A: No domain randomization. B: Randomization of position control PID parameters and friction. C: Same as B, but also including the previous action as part of the state space.

We see in Fig. 19 that the randomization of PID parameters and friction is crucial for effective transfer.

While simulation to real transfer enabled by randomization is an appealing option, especially for fragile robots, it has a number of limitations. First, the resulting policies can end up being overly conservative due to the randomization, a phenomenon that has been widely observed in the field of robust control. Second, the particular choice of parameters to randomize is crucial for good results, and insights from one task or problem domain may not transfer to others. Third, increasing the amount of randomization results in more complex models tremendously increasing the training time and required computational resources, as discussed in Section II. Directly training in the real world may be more efficient and lead to better policies. Finally, and perhaps most importantly, an accurate simulator must be constructed manually, with each new task modeled by hand in the simulation, which requires substantial time and expertise. For tasks such as valve rotation with the foam valve or door opening with a soft handle, creating the simulation itself is very challenging.

E. Design Choices

To understand the design choices needed to effectively train dexterous hand manipulation systems with model-free RL, we analyzed different factors which contribute to learning. We performed this analysis in simulation in order to choose the right schemes for real world training.

1) Choices of Action Space: The choice of actuation space often makes an impact on learning progress . For hand manipulation it also greatly affects the smoothness, and hence sustainability, of the hardware. In our results, we end up using position control since it induces the fewest vibrations and is easiest to learn with.

In order to better understand the rationale behind this, we consider a comparison between using controlling position and torque controllers as well as their higher order derivatives. We compare the vibrations induced by each of these control schemes by measuring the sum of the magnitudes of the highest Fourier coefficients of sample trajectories (joint angles) induced by random trajectories.

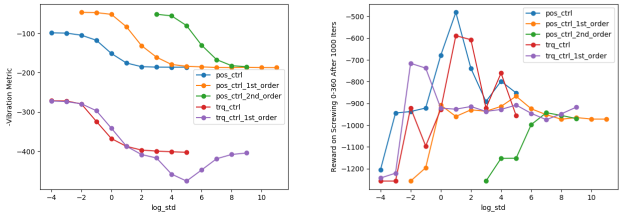

Fig. 20: Left: Analysis of vibrations induced by different actuation schemes in simulation. Higher metric indicates lower vibrations. We find that position control is able to induce significantly lower vibrations than torque control, making it safer to run on hardware. Right: Analysis of rewards attained in simulation after training using the control scheme on Dclaw valve rotating task.

We see that position control has the lowest vibration amongst all the choices of control schemes, and is also able to achieve significantly better performance. This is likely because we are using a stabilizing PID control at the low level to do position control which reduces the load on the learning algorithm. We also see that it is harder to learn a policy when controlling higher order derivatives, and that it is easier to learn with position control than with torque control.

2) Impact of Reward Function: We also investigated the effect of the reward function on learning progress. To provide some intuition about the different choice of reward functions, we show a comparison between 3 different reward functions for learning. We evaluate learning progress for these three types of reward functions in simulation, as a means for choosing an appropriate form of reward for real world training.

1) $r_1 = -\|\theta - \theta_{goal}\|_2$
2) $r_2 = r_1 + 10 * \mathbb{1}_{\{r_1<0.1\}} + 50 * \mathbb{1}_{\{r_1<0.05\}}$
3) $r_3 = r_2 - \|v\|_2$

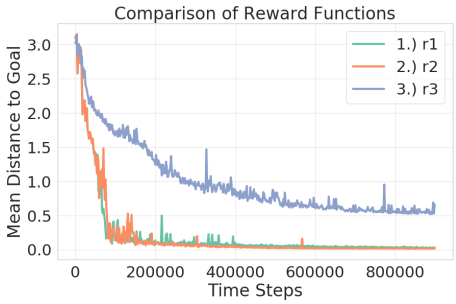

Fig. 21: Analysis of learning progress in simulation with different reward functions.

We find that learning is most effective with using either option 1 or 2, and is slower with a control cost. The control cost ensures smoother operation, but at the cost of efficiency in learning, since optimization of control penalties results in reduction of exploration. In real world experiments we

found that training without a control cost still produced safe behaviors.

VII. Discussion and Future Work

In this work, we study real-world model-free reinforcement learning as a means to learn complex dexterous hand manipulation behaviors. We show that model-free deep RL algorithms can provide a practical, efficient, and general method for learning with high dimensional multi-fingered hands. Model-free RL algorithms can easily be applied to a variety of low-cost hands, and solve challenging tasks that are hard to simulate accurately. This kind of generality and minimal manual engineering may be a key ingredient in endowing robots with the kinds of large skill repertoires they need to be useful in open-world environments such as homes, offices, and hospitals. We also show that the sample complexity of model-free RL can be substantially reduced with suboptimal kinesthetic demonstrations, while also improving the resulting motion quality.

There are several exciting avenues for future work. Firstly, deep neural networks can not only represent complex skills, but can also high-dimensional inputs such as images. Studying dexterous manipulation with visual observations is an exciting direction for future work. Furthermore, while we learn each skill individually in our experiments, an exciting future direction is to pool experience across multiple skills and study the ability to acquire new behaviors more efficiently in a multi-task setting, which is an important stepping stone toward generalist robots endowed with large behavioral repertoires.

References

[1] A. Rajeswaran, V. Kumar, A. Gupta, J. Schulman, E. Todorov, and S. Levine, "Learning complex dexterous manipulation with deep reinforcement learning and demonstrations," *CoRR*, vol. abs/1709.10087, 2017.

[2] V. Kumar, Z. Xu, and E. Todorov, "Fast, strong and compliant pneumatic actuation for dexterous tendon-driven hands," in *2013 IEEE International Conference on Robotics and Automation, Karlsruhe, Germany, May 6-10, 2013*, pp. 1512–1519, 2013.

[3] J. Butterfaß, M. Grebenstein, H. Liu, and G. Hirzinger, "Dlr-hand II next generation of a dextrous robot hand," in *Proceedings of the 2001 IEEE International Conference on Robotics and Automation, ICRA 2001, May 21-26, 2001, Seoul, Korea*, pp. 109–120, 2001.

[4] A. Bicchi and V. Kumar, "Robotic grasping and contact: A review," in *ICRA*, vol. 348, p. 353, Citeseer, 2000.

[5] A. M. Okamura, N. Smaby, and M. R. Cutkosky, "An overview of dexterous manipulation," in *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, vol. 1, pp. 255–262, IEEE, 2000.

[6] I. M. Bullock, R. R. Ma, and A. M. Dollar, "A hand-centric classification of human and robot dexterous manipulation," *IEEE transactions on Haptics*, vol. 6, no. 2, pp. 129–144, 2013.

[7] X. Zhu and J. Wang, "Synthesis of force-closure grasps on 3-d objects based on the q distance," *IEEE Transactions on robotics and Automation*, vol. 19, no. 4, pp. 669–679, 2003.

[8] A. T. Miller and P. K. Allen, "Graspit! a versatile simulator for robotic grasping," *IEEE Robotics & Automation Magazine*, vol. 11, no. 4, pp. 110–122, 2004.

[9] R. M. Murray, *A mathematical introduction to robotic manipulation*. CRC press, 2017.

[10] I. Mordatch, Z. Popović, and E. Todorov, "Contact-invariant optimization for hand manipulation," in *Proceedings of the ACM SIGGRAPH/Eurographics symposium on computer animation*, Eurographics Association, 2012.

[11] Y. Bai and C. K. Liu, "Dexterous manipulation using both palm and fingers," in *ICRA 2014*, IEEE, 2014.

[12] V. Kumar, Y. Tassa, T. Erez, and E. Todorov, "Real-time behaviour synthesis for dynamic hand-manipulation," in *2014 IEEE International Conference on Robotics and Automation, ICRA 2014, Hong Kong, China, May 31 - June 7, 2014*, pp. 6808–6815, 2014.

[13] V. Kumar, E. Todorov, and S. Levine, "Optimal control with learned local models: Application to dexterous manipulation,"

[14] A. Gupta, C. Eppner, S. Levine, and P. Abbeel, "Learning dexterous manipulation for a soft robotic hand from human demonstrations," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2016, Daejeon, South Korea, October 9-14, 2016*, pp. 3786–3793, 2016.

[15] H. van Hoof, T. Hermans, G. Neumann, and J. Peters, "Learning robot in-hand manipulation with tactile features," in *Humanoid Robots (Humanoids)*, IEEE, 2015.

[16] J. Kober and J. Peters, "Policy search for motor primitives in robotics," in *Advances in Neural Information Processing Systems 21, Proceedings of the Twenty-Second Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 8-11, 2008*, pp. 849–856, 2008.

[17] A. J. Ijspeert, J. Nakanishi, and S. Schaal, "Learning attractor landscapes for learning motor primitives," tech. rep., 2002.

[18] M. Vecerik, T. Hester, J. Scholz, F. Wang, O. Pietquin, B. Piot, N. Heess, T. Rothörl, T. Lampe, and M. A. Riedmiller, "Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards," *CoRR*, vol. abs/1707.08817, 2017.

[19] A. Nair, B. McGrew, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Overcoming exploration in reinforcement learning with demonstrations," *CoRR*, vol. abs/1709.10089, 2017.

[20] T. Hester, M. Vecerik, O. Pietquin, M. Lanctot, T. Schaul, B. Piot, D. Horgan, J. Quan, A. Sendonaris, I. Osband, G. Dulac-Arnold, J. Agapiou, J. Z. Leibo, and A. Gruslys, "Deep q-learning from demonstrations," in *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, New Orleans, Louisiana, USA, February 2-7, 2018*, 2018.

[21] Y. Zhu, Z. Wang, J. Merel, A. A. Rusu, T. Erez, S. Cabi, S. Tunyasuvunakool, J. Kramár, R. Hadsell, N. de Freitas, and N. Heess, "Reinforcement and imitation learning for diverse visuomotor skills," *CoRR*, vol. abs/1802.09564, 2018.

[22] A. Rajeswaran, S. Ghotra, S. Levine, and B. Ravindran, "Epopt: Learning robust neural network policies using model ensembles," *CoRR*, vol. abs/1610.01283, 2016.

[23] F. Sadeghi and S. Levine, "(cad)^{2}rl: Real single-image flight without a single real image," *CoRR*, vol. abs/1611.04201, 2016.

[24] S. James, A. J. Davison, and E. Johns, "Transferring end-to-end visuomotor control from simulation to real world for a multi-stage task," in *Conference on Robot Learning (CoRL)*, 2017.

[25] F. Sadeghi, A. Toshev, E. Jang, and S. Levine, "Sim2real view invariant visual servoing by recurrent control," *CoRR*, vol. abs/1712.07642, 2017.

[26] J. Tan, T. Zhang, E. Coumans, A. Iscen, Y. Bai, D. Hafner, S. Bohez, and V. Vanhoucke, "Sim-to-real: Learning agile locomotion for quadruped robots," *CoRR*, vol. abs/1804.10332, 2018.

[27] OpenAI, "Learning dexterous in-hand manipulation," *CoRR*, vol. abs/1808.00177, 2018.

[28] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine Learning*, vol. 8, pp. 229–256, May 1992.

[29] J. Schulman, S. Levine, P. Abbeel, M. I. Jordan, and P. Moritz, "Trust region policy optimization," in *Proceedings of the 32nd International Conference on Machine Learning, ICML*, 2015.

[30] J. Peters and S. Schaal, "Natural actor-critic," *Neurocomputing*, vol. 71.

[31] A. Rajeswaran, K. Lowrey, E. V. Todorov, and S. M. Kakade, "Towards generalization and simplicity in continuous control," in *NIPS*, 2017.