# Bootstrapping Motor Skill Learning with Motion Planning

Ben Abbatematteo*, Eric Rosen*, Stefanie Tellex, George Konidaris
Department of Computer Science
Brown University, Providence RI
{babbatem,er35,stefie10,gdk}@cs.brown.edu

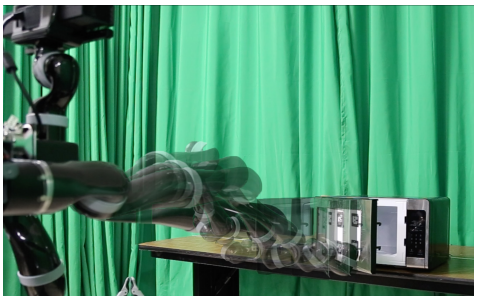
***Abstract*— Learning a robot motor skill from scratch is impractically slow; so much so that in practice, learning must typically be bootstrapped using human demonstration. However, relying on human demonstration necessarily degrades the autonomy of robots that must learn a wide variety of skills over their operational lifetimes. We propose using kinematic motion planning as a completely autonomous, sample efficient way to bootstrap motor skill learning for object manipulation. We demonstrate the use of motion planners to bootstrap motor skills in two complex object manipulation scenarios with different policy representations: opening a drawer with a dynamic movement primitive representation, and closing a microwave door with a deep neural network policy. We also show how our method can bootstrap a motor skill for the challenging dynamic task of learning to hit a ball off a tee, where a kinematic plan based on treating the scene as static is insufficient to solve the task, but sufficient to bootstrap a more dynamic policy. In all three cases, our method is competitive with human-demonstrated initialization, and significantly outperforms starting with a random policy. This approach enables robots to to efficiently and autonomously learn motor policies for dynamic tasks without human demonstration.**

## I. INTRODUCTION

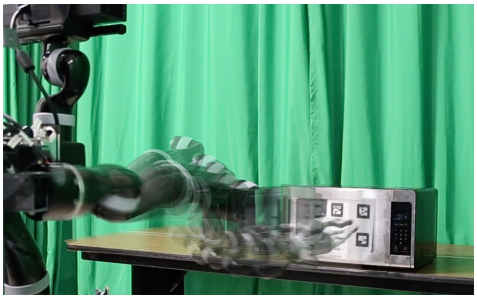
Robots require motor policies for interacting with objects in their environment. Reinforcement learning (RL) provides a framework for acquiring motor policies without explicitly modeling the unknown world, but model-free RL methods like policy search [1] have high sample-complexity, and often fail to learn a reasonable policy from random initialization. Supervised approaches for policy learning like Learning From Demonstration (LfD) [2] can encode human prior knowledge by imitating expert examples, but do not support optimization in new environments. Combining RL with LfD is a powerful method for reducing the sample complexity of policy search, and is often used in practice [3, 4, 5, 6]. However, this approach typically requires a human demonstrator for initialization, which fundamentally limits the autonomy, and therefore utility, of a robot that may need to acquire a wide range of motor skills over its operational lifetime. More recently, model-based control techniques (including Model Predictive Control [7] and LQR [3]) have been proposed as exploration methods for policy search; these methods still require human demonstrations or complete dynamic models of both the robot and every object in the scene.

We propose the use of kinematic motion planning to initialize motor skill policies. While previous work has leveraged sample-based motion planners for learning motor skills [8, 9, 10], they only focus on either free-space motions or do not learn a closed-loop controller. To our knowledge, this is the first use of motion planning to provide initial demonstrations for learning closed-loop motor skill policies by leveraging estimated object kinematics.

We show that given a (potentially approximate, and readily estimated) kinematic description of the environment and the robot, off-the-shelf motion planning algorithms can generate feasible (potentially successful but inefficient) initial trajectories (Figure 1a) to bootstrap an object-manipulation policy that can subsequently be optimized using policy search (Figure 1b). This framework enables the robot to automatically



(a)



(b)

Fig. 1: A robot using our method to autonomously learn to close a microwave that is out of reach. (a) The robot uses a motion planner to generate an initial attempt at closing the microwave door using a kinematic model of the microwave. The resulting plan is unable to fully close the microwave door because of the robot's limited reach. (b) After bootstrapping a motor skill with the trajectory from (a), the robot learns a motor skill that gives the door a push, exploiting its dynamics to fully close the microwave.

*These authors contributed equally.

produce its own demonstrations for effectively learning and refining object manipulation policies. Our work enables the robot to exploit kinematic planning to realize the benefits of an initial demonstration fully autonomously.

To evaluate our method, we used two different motor policy classes (Dynamic Movement Primitives (DMPs) [11] and deep neural networks [12]). We compared bootstrapping with motion planning against learning from scratch in three simulated experiments, and against human demonstrations in real hardware experiments. We show that motion planning using a kinematic model produces a reasonable, though suboptimal, initial policy compared to a supervised human demonstration, which learning adapts to generate efficient, dynamic policies that exploit the dynamics of the object being manipulated. Our method is competitive with human-demonstrated initialization. It serves as a suitable starting point for learning, and significantly outperforms starting with a random policy. Taken together, these results show that our method is competitive with human demonstrations as a suitable starting point for learning, enabling robotics to efficiently and autonomously learn motor policies for dynamic tasks without human demonstration.

## II. BACKGROUND

Our goal is to efficiently and autonomously learn robot motor skill policies. To do so, we develop an approach that uses kinematic motion planning to generate initial trajectories, fits a policy to those trajectories using behavioral cloning, and subsequently optimizes that policy via policy search. We now briefly describe policy search, policy representations, learning from demonstration, and motion planning.

### A. Policy Search

Policy search methods [1] are a family of model-free reinforcement learning algorithms that search within a parametric class of policies to maximize reward. Formally, given a Markov Decision Process $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{T}, \gamma \rangle$, the objective of policy search is to maximize the expected return of the policy $\pi_\theta$:

$$\max_\theta \mathop{\mathbb{E}}_{\mathcal{M},\pi_\theta} \left[ \sum_{t=0}^{T} \gamma^t r_t \right]. \tag{1}$$

These approaches can learn motor skills through interaction and therefore do not require an explicit environment model, and are typically agnostic to the choice of policy class (though their success often depends on the policy class having the right balance of expressiveness and compactness). However, their model-free nature leads to high sample complexity, often making them infeasible to apply directly to robot learning.

**Policy representation** describes the class of functions used as the mapping from states to actions; we use two common representations in this paper (Dynamic Movement Primitives and neural networks), but our method is class agnostic.

Dynamic Movement Primitives [11] are a description of a non-linear second-order differential equation that exhibits attractor dynamics modulated by a learnable forcing function. DMPs are a popular representation for motor policies because they are parameter-efficient —$n$ joints are controlled using $n$ independent DMPs coupled only through time, which is very compact— can express both point and limit cycle attractors, enable real-time computation, and exhibit temporal invariance that does not effect the attractor landscape. We refer the reader to Ijspeert et al. [11] for a more formal introduction.

Neural network controllers have received significant attention in recent years; they are able to learn hierarchical feature representations for approximating functions (in our case, motor skills) operating on high-dimensional input such as robot sensor data [12]. They are more expressive than restricted policy classes such as DMPs and can operate directly on high-dimensional state spaces (e.g. images), but typically exhibit high sample complexity.

We chose these two different motor policy classes because they represent opposite ends of the policy spectrum: deep neural networks are extremely expressive in what policies they can represent, but are extremely sample inefficient compared to structured motor primitives like DMPs, which are more structured and less expressive.

**Learning from Demonstration** methods [2] broadly consist of two families of approaches that either mimic (Behavioral Cloning) or generalize (Inverse Reinforcement Learning) demonstrated behavior. Inverse reinforcement learning methods estimate a latent reward signal from a set of demonstrations; we assume a given reward function, and omit a discussion of inverse reinforcement learning methods here.

Behavioral cloning methods [13, 14, 15] attempt to directly learn a policy that reproduces the demonstrated data. Given a dataset of expert demonstrations $D$, the objective of behavioral cloning is: $\max_\theta \sum_{(s,a)\in D} \pi_\theta(a|s)$. These methods often result in policies with undesirable behavior in states not observed during demonstrations, though this can addressed with interactive learning [16, 17, 18]. In our approach, the existence of a reward function enables the agent to learn robust behavior outside of the initial training distribution. Moreover, our experiments demonstrate our approach's ability to extrapolate beyond suboptimal initial demonstrations.

Many approaches investigate incorporating human-provided demonstrations into policy search to drastically reduce sample complexity via a reasonable initial policy and/or the integration of demonstrations in the learning objective [5, 4, 3]. Approaches like these (and ours) are complimentary to exploration strategies for behavioral policies during policy learning.

### B. Motion Planning

The pose of an articulated rigid body can defined by the state of its movable joints. The space of these poses is called the configuration space $\mathcal{C}$ [19]. Motion planning is the problem of finding a path (sequence of poses) through

configuration space such that the articulated object is moved to a desired goal configuration, without encountering a collision.

While there exist many different families of motion planning algorithms, such as geometric, grid-based, and probabilistic road maps [20], they all operate in a similar fashion: given a configuration space $\mathcal{C}$ and start and goal joint configurations $q_0, q^* \in \mathcal{C}$, return a valid path of joint configurations $\{q_t\}_{t=0}^{T}$ between the start and end configurations.

Probabilistic motion planners provide a principled approach for quickly generating collision-free robot trajectories. However, online replanning is expensive, and kinematic motion planners are only as effective as their kinematic models are accurate: they generate trajectories directly, and thus cannot be improved through subsequent interaction and learning. Furthermore, kinematic planners typically produce trajectories that only account for kinematics, not dynamics: they explicitly do not account for forces involved in motion, such as friction, inertial forces, motor torques, etc, which are important for effectively performing contact-rich, dexterous manipulation.

The process of computing the position and orientation $p \in SE(3)$ of a link in a kinematic chain for a given joint variable setting (a point in configuration space) is termed *forward kinematics*. Inversely, computing a configuration to attain a specific end effector pose $p$ is termed *inverse kinematics*. We denote the forward kinematics functions $p = f(q)$.

## III. Bootstrapping Skills with Motion Planning

Our methodology is inspired by how humans generate reasonable first attempts for accomplishing new motor tasks. When a human wants to learn a motor skill, they do not start by flailing their arms around in a random fashion, nor do they require another person to guide their arms through a demonstration. Instead, they make a rough estimate of how they want an object to move and then try to manipulate it to that goal. For example, before being able to drive stick shift, a human must first learn how to manipulate a gear shifter for their car. Just by looking at the gear shifter, humans can decide (1) what they should grab (the shaft), (2) where they want the shaft to go (positioned in a gear location), and (3) how the shaft should roughly move throughout the action (at the intermediate gear positions). Similarly, a robot that has a good kinematic model of itself, and a reasonable kinematic model of the object it wishes to manipulate, should be able to form a motion plan to achieve the effect it wishes to achieve.

That plan may be inadequate in several ways: its kinematic model may be inaccurate, so the plan does not work; object dynamics (like the weight of a door, or the friction of a joint) may matter, and these are not represented in a kinematic model; or a feasible and collision-free kinematic trajectory may not actually have the desired effect when executed on a robot interacting with a real (and possibly novel) object. These are all the reasons why a novice driver can immediately shift gears, but not very well. But such a solution is a *good start*; we therefore propose to use it to bootstrap motor skill learning.

Our approach, outlined in Figure 2, leverages the (partial) knowledge the robot has about its own body and the object it is manipulating to bootstrap motor skills. Our method first assumes access to the configuration space of the robot, denoted as $\mathcal{C}_R$, as well as its inverse kinematics function $f_R^{-1}$. This assumption is aligned with the fact that the robot often has an accurate description of its own links and joints and how they are configured during deployment. However, the world is comprised of objects with degrees of freedom that can only be inferred from sensor data. Therefore, our approach only assumes access to estimated kinematics of the object to be manipulated, in the form of configuration space $\mathcal{C}_O$ and forward kinematics $f_O$. Recent work has shown that estimating these quantities for novel objects from sensor data in real environments is feasible [21, 22], though state-of-the-art estimates still include noise.

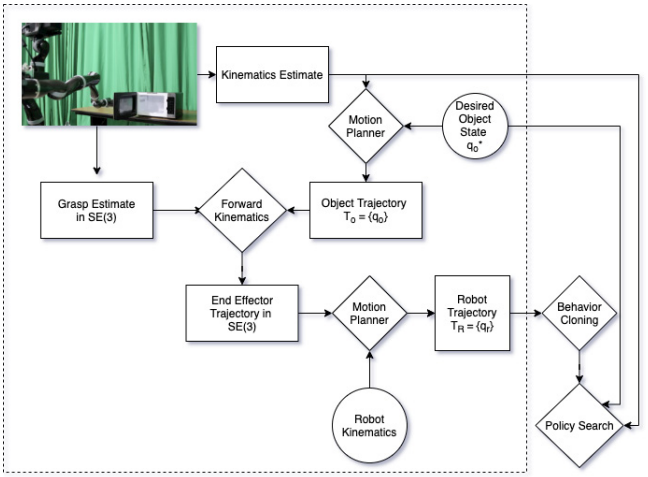
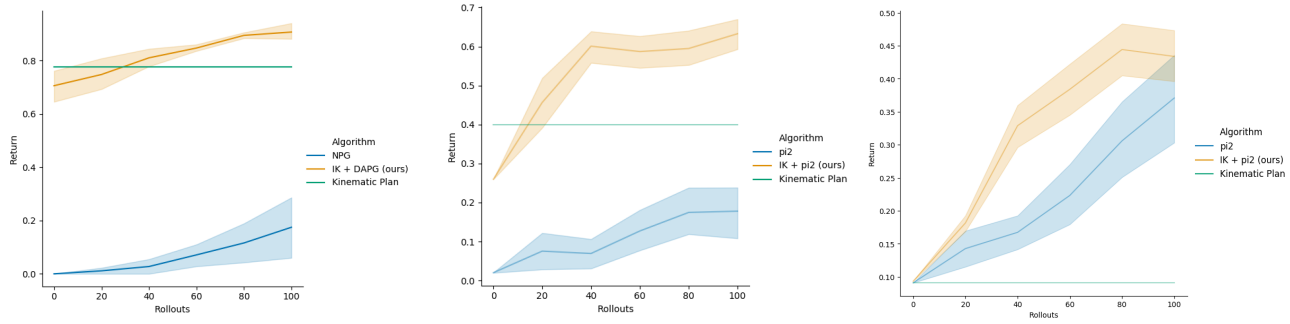



Fig. 2: **System overview** illustrating our proposed framework for generating demonstrations with a motion planner and subsequently performing policy search. The dashed box contains the steps from Algorithm 2.

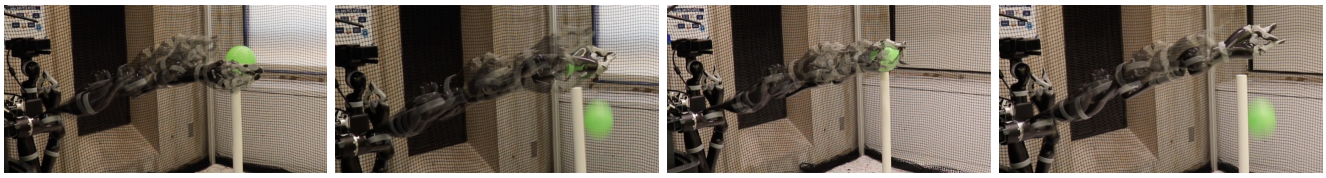
Finally, our approach assumes that the task goal can be defined in terms of kinematic states of the robot and environment. Examples of such tasks include pick-and-place, articulated object manipulation, and many instances of tool use. (Note that this requirement cannot capture reward functions defined in terms of force, for example exerting a specific amount of force in a target location.) Such a goal, together with object and robot kinematics, enables us to autonomously generate useful initial trajectories for policy search.

Our approach is outlined in Algorithm 1, and can broken down into five main steps: 1) collect initial trajectories from a motion planner using estimated object kinematics, 2) fit a policy with these initial trajectories, 3) gather rollouts to sample rewards for the current policy based on the kinematic goal, 4) update the policy parameters based on the actions and rewards, 5) repeat steps 3-4.




(a) Microwave closing (MLP) (b) Drawer opening (DMP) (c) T-ball (DMP)

Fig. 3: **Simulation Results.** a) Comparison of our method optimized with DAPG against Natural Policy Gradient starting with a random policy in a microwave closing task using Gaussian multi-layer perception policies. b) Comparison of our method against PI$^2$-CMA starting with a random policy in a drawer opening task with DMP policies. c) Our method compared with PI$^2$-CMA with a initially random policy in t-ball with DMP policies. Results are shown as mean and standard error of the normalized returns aggregated across 20 random seeds.



(a) Human Demonstration (b) Bootstrapped from (a) (c) Motion Plan Demonstration (d) Bootstrapped from (c)

Fig. 4: **Real-world Ball Hitting** Images comparing bootstrapping motor skills with a human demonstration vs. a motion planner on a real-world robot hitting a ball off a tee. In both cases, the bootstrapped motor skill outperforms the initial demonstration. Videos can be found in our supplemental video. (a) A demonstration provided by a human teleoperating the robot. (b) A motor skill bootstrapped by the human demonstration. (c) A demonstration provided by a motion planner. (d) A motor skill bootstrapped by the motion planner demonstration.

**Algorithm 1** Planning for Policy Bootstrapping

**procedure** PPB($C_R, f_R^{-1}, C_O, f_O, q_O^*$)
  $D \leftarrow \varnothing$
  **for** 0 to $N$ **do**
    $D \leftarrow D \cup$ InitialMPDemos($C_R, f_R^{-1}, C_O, f_O, q_O^*$)
  **end for**
  $\theta \leftarrow$ FitPolicy($D_0, ..., D_N$)
  **for** 0 to $E$ **do**
    $T_0, .., T_n \leftarrow$ Rollout($\pi, \theta, q_O^*$)
    $\theta \leftarrow$ UpdatePolicy($T_1, .., T_n, \theta$)
  **end for**
**end procedure**

**Algorithm 2** Initial Motion Plan Demos

**procedure** INITIALMPDEMOS($C_R, f_R^{-1}, C_O, f_O, q_O^*$)
  $T_O \leftarrow$ MotionPlanner($C_O, q_O^*$)
  $g \leftarrow$ EstimateGrasp($C_O$,$f_O$)
  $eepath \leftarrow$ GraspPath($T_O, C_O, f_O, g$)
  $T_R \leftarrow$ MotionPlanner($C_R$, $eepath$, $f_R^{-1}$)
  return $T_R$
**end procedure**

### A. *Fitting a Policy to a Demonstration*

After collecting initial demonstrations from the motion planner, $D$, we can bootstrap our motor policy by initializing the parameters to the policy $\theta$ using any behavioral cloning technique; in practice, we use Locally Weighted Regression [23] for DMPs, and maximize the likelihood of the demonstration actions under the policy for neural networks.

### B. *Policy Search with Kinematic Rewards*

To improve the motor policies after bootstrapping, we can perform policy search based on the given (kinematic) reward function. Specifically, we choose a number of epochs $E$ to perform policy search for. For each epoch, we perform an iteration of policy search by executing the policy and collecting rewards based on the goal $q_O^*$. We define our reward functions using estimated object states $q_O$ and goal states $q_O^*$, and add a small action penalty.

## IV. EXPERIMENTS

The aim of our evaluation was to test the hypothesis that motion planning can be used to initialize policies for learning from demonstration without human input. We tested this hypothesis in simulation against learning from scratch, and on real hardware, against human demonstrations, on three tasks: microwave-closing, drawer-opening, and t-ball.

We note that we do not show asymptotic performance because our emphasis is on learning on real hardware from a practical number of iterations. All the elements of the motion planner—state sampler, goal sampler, distance metrics, etc.—are reused between problems without modification.

### A. Simulation Experiments

We used PyBullet [24] to simulate an environment for our object manipulation experiments. We used URDFs to instantiate a simulated 7DoF KUKA LBR iiwa7 arm and the objects to be manipulated, which gave us ground-truth knowledge of the robot and object kinematics. For all our simulated experiments, we compared implementations of our method against starting with a random policy.

For all three tasks, the state was represented as $s_t = [q_R, q_O]^T$ where $q_R$ denotes robot configuration and $q_O$ denotes object configuration. The action space $\mathcal{A}$ was commanded joint velocity for each of the 7 motors. The reward at each timestep $r_t$ was given as:

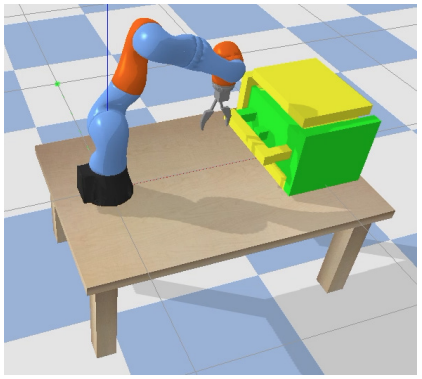
$$r_t = -c \; ||q_O^* - q_O||_2^2 - a_t^T R a_t, \tag{2}$$

where $q_O$ denotes the object state at time $t$, $q_O^*$ denotes desired object state, and $a_t$ denotes the agent's action. We set $c = 60$ and R = $I \times 0.001$ for all experiments. As such, maximum reward is achieved when the object is in the desired configuration, and the robot is at rest.

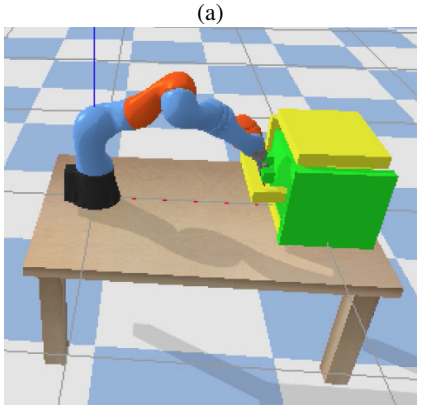
Our first simulated task was to close a microwave door, which consisted of three parts: a base, a door, and a handle. The pose of the handle was used for the EstimateGrasp method in Algorithm 2. The robot was placed within reaching distance of the handle when the microwave door was in an open position, but was too far to reach the handle in its closed configuration. Thus, the agent was forced to push the door with enough velocity to close it. We used Gaussian policies represented as multi-layer perceptrons with two hidden layers of sizes (32,32) in this experiment. The randomly initialized policy was optimized with natural policy gradient [25]. Ten demonstrations were generated by perturbing the start state and initial kinematic plan with Gaussian noise. The behavior cloning was performed by maximizing likelihood over the demonstration dataset for 10 epochs. Our pretrained policy was optimized using Demo Augmented Policy Gradient [4], which essentially adds the behavior cloning loss to the natural policy gradient loss, annealing it over time. This ensures that the agent remains close to the demonstrations early in learning, but is free to optimize reward exclusively as learning progresses. Results are shown in Figure 3a.

The second simulated task was to open a drawer (Figure 5). This task required the agent to grasp the drawer's handle and pull the drawer open.

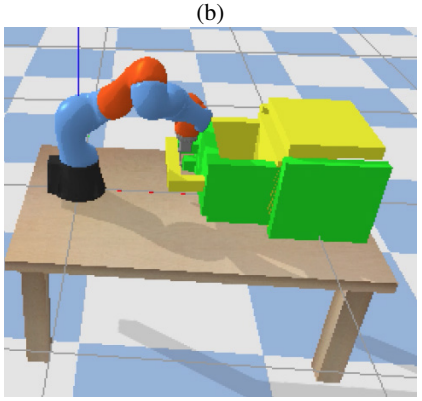
Again, the pose of the object's handle was used for EstimateGrasp method in our algorithm. In this experiment, we used DMP policies. The weights, goals, and speed parameters of the policies were optimized using PI$^2$-CMA [26]. We used 32 basis functions for each of the DMPs. The pretrained policy was initialized using Locally Weighted



(a)



(b)



(c)

Fig. 5: **Opening a Drawer** experiment in simulation, where the robot needs to apply enough force on the handle to slide the drawer open. (a) An image of the starting pose of the robot arm and drawer. When learning from scratch, the robot random explores for many steps before grasping the handle. (b) An image of the robot using our method to produce an initial demonstration from a motion planner based on the drawer's kinematics. This demonstration guides the robot to the handle, but ignores the dynamics of the heavy drawer which leads to failure. (c) An image after the robot has bootstrapped a skill with our method. The final policy learns to leverage the dynamics to precisely grasp the handle and then produce a strong pulling force to open the drawer completely.

Regression (LWR) [23] with a single demonstration. The results of this experiment are shown in Figure 3b.

The third simulated task was to hit a ball off a tee. The ball started at rest on top of the tee. The pose of the ball was used in the EstimateGrasp method. The object state was defined as the object's $y$ position relative to its initial pose. This is a poor initialization for a hitting task because it is based only on the ball's kinematics and ignores the dynamics involved in swinging, resulting in low-return, but it is effective for bootstrapping policy search. This experiment again used DMPs initialized with LWR and optimized with PI$^2$-CMA. Results of this experiment are visualized in Figure 3c.

Across all three tasks, we observe that policies initialized with our method dramatically outperform starting learning with a random policy. This confirms our hypothesis that using motion planning to generate demonstrations significantly speeds the acquisition of motor skills in challenging tasks like articulated object manipulation and t-ball.

### B. Real-world Experiments

For all our real-world experiments, we used a 7DoF Jaco arm [27] to manipulate objects (Figure 1). We used ROS and MoveIt! [28] as the interface between the motion planner (RRT* [29] in our experiments) and robot hardware. For all real-world experiments, we compared implementations of our method against bootstrapping with a human demonstration, which we supplied.* To collect human demonstrations, we had an expert human teleoperate the robot with joystick control to perform the task. For all tasks, the state space, action space, and reward were defined in the same way as in our simulated results (Section IV-A). Both experiments used DMP policies initialized with LWR [23] and optimized with PI$^2$-CMA [26] with 10 basis functions for each of the DMPs.

Our first real-world task was to close a microwave door, similar to the one described in our simulated domain (Section IV-A). As in the simulated microwave task, we used the pose of the handle for the EstimateGrasp method in Algorithm 2, and also the robot was similarly placed such that it was forced to push the door with enough velocity to close. We placed an AR tag on the front-face of the microwave to track the microwave's state using a Kinect2. Results are shown in Figure 6.

We observe that the human demonstration is better than the one produced by the motion planner, which we credit to the fact that the motion of the door was heavily influenced by the dynamics of the revolute joint which the motion planner did not account for. Nonetheless, both policies converge to a similar final performance, with our method converging slightly faster. Note the importance of the policy search: the motion planner alone is insufficient for performing the task efficiently.

*We acknowledge this potential bias in expert trajectories, and qualify our decision by only training on human demonstrations that at least accomplished the task.

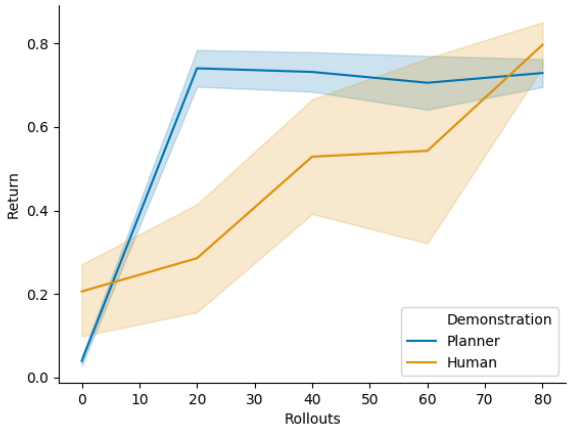



Fig. 6: **Hardware experiment** comparing our initialization scheme with human demonstration. Results are shown as mean and standard error, aggregated across three random seeds.

Our second real-world task was to hit a ball off a tee as far as possible (Figure 4). Similar to our simulated task, the ball started at rest on top of the tee. The pose of the ball was used in the EstimateGrasp method. The object state was defined as the object's $y$ position relative to its initial pose. We placed scotchlite-reflective tape on the surface of the ball and conducted our experiments within an OptiTrack motion-capture cage to track the object pose. We observe that when using a motion planner to hit the ball, it moves the bat in a linear motion to make contact, therefore transferring only horizontal motion to the ball. We qualitatively observe that during policy search, the robot learns a dynamic policy that accounts for the dynamics of the ball by applying force under the ball to "scoop" the ball upwards and forwards.

## V. RELATED WORK

To our knowledge, our method is the first to use an object's estimated kinematics in conjunction with a known robot model to bootstrap motor policy learning.

While classic robot motor learning papers [13] leverage the known kinodynamics of the robot, they do not discuss kinematics of external objects or grasp candidates to bootstrap motor policies for object manipulation. We emphasize that we cannot form dynamic plans in the problem setting we are interested in: objects with unknown a priori dynamics.

Recently, Model-Predictive Control (MPC) has been used in the context of imitation learning and reinforcement learning to address the high sample complexity of policy search [7]. These approaches require a priori object dynamics, or human demonstrations to fit learned models; in contrast, our approach requires only object kinematics, which are much more readily estimated from visual data at runtime [21, 22]. As such, our approaches enables the learning of manipulation skills to be more autonomous than existing MPC-based methods. Tosun et al. [8] proposed a neural network model for

generating trajectories from images, using a motion planner during training to enable the robot to generate a trajectory with a single forward pass at runtime. While this approach uses a motion planner for behavior cloning, it stops short of optimization to improve the resulting policy. By contrast, our method uses object kinematics to produce initial trajectories, while Tosun et al. [8] only use the robot's kinematic model, which is insufficient when the task is to manipulate an object to a specific joint configuration.

Kurenkov et al. [30] proposed training an initially random RL policy with an ensemble of task-specific, hand-designed heuristics. This improves learning but the initial policy is still random, yielding potentially unsafe behavior on real hardware, and delaying convergence to a satisfying policy. By contrast, we choose to initialize the policy with demonstrations from a kinematic planner, ensuring feasibility, safety, and rapid learning. Moreover, we argue that motion planning is the principled heuristic to use to accelerate learning, as it is capable of expressing manually programmed heuristics like reaching and pulling. Finally, our approach can use the existing estimated object kinematics to provide a principled reward signal for model-free reinforcement learning.

Recently, residual reinforcement learning approaches have been developed which learn a policy superimposed on hand-designed or model-predictive controllers [31, 32]. Our method is compatible with these approaches, where demonstrations from the motion planner can be used as a base policy on top of which a residual policy can be learned based on kinematic rewards. These methods typically suffer from the same limitations as MPC-based methods mentioned above.

Guided Policy Search (GPS) [3] uses LQR to guide policy search into high-reward regions of the state-space. The models employed are fundamentally local approximations, and thus would benefit greatly from a wealth of suboptimal demonstrations from the outset (as made evident by Chebotar et al. [33]). GPS is one of the state-of-the-art algorithms we expect to be used within our framework as the policy search implementation (Section III-B). A critical distinction between our work and GPS is the notion of planning trajectories in object configuration spaces and reasoning about grasp candidates to achieve a desired manipulation. This is done using information available a priori, and thus is immediately capable of generating high-value policies, whereas GPS estimates dynamics models given observed data (obtained either from demonstration or random initialization). In the absence of a human demonstrator, our method would provide far more useful data at the outset of learning than running a naively initialized linear-Gaussian controller (as evidenced by our comparisons to random initialization).

Most similar to our line of work are those that use sample-based motion planners for improved policy learning. Jurgenson and Tamar [9] harness the power of reinforcement learning for neural motion planners by proposing an augmentation of Deep Deterministic Policy Gradient (DDPG) [34] that uses the known robot dynamics to leverage sampling methods like RRT* to reduce variance in the actor update and provide off-policy exploratory behavior for the replay buffer. However, Jurgenson and Tamar [9] are only able to address domains where they can assume good estimates of the dynamics model, such as producing free-space motions to avoid obstacles. Our setting, in contrast, focuses on object manipulation, where dynamics are not readily available, but are critical for learning good policies. Jiang et al. [10] address learning to improve plans produced by a motion planner, but do not bootstrap closed-loop policies. Motion planners are insufficiently expressive to leverage the dynamics in object-manipulation tasks, especially in the presence of unknown dynamics, and traditionally are unable to handle perceptual data like RGB images. Our method, on the other hand, enables motion planning to bootstrap policies that are more expressive than the original planner.

## VI. CONCLUSION

We have presented a method that uses kinematic motion planning to bootstrap robot motor policies. By assuming access to a potentially noisy description of the object kinematics, we are able to autonomously generate initial demonstrations that perform as well as human demonstrations—but do not require a human—resulting in a practical method for autonomous motor skill learning.

Our methodology is agnostic to the motion planner, motor policy class, and policy search algorithm, making it a widely applicable paradigm for learning robot motor policies. We demonstrate the power of our methodology by bootstrapping different policy classes with demonstrations from humans and a motion planner, and learn motor policies for three dynamic manipulation tasks: closing a microwave door, opening a drawer, and hitting a ball off a tee. Our framework is the first to enable robots to autonomously bootstrap and improve motor policies with model-free reinforcement learning using only a partially-known kinematic model of the environment. Future work will incorporate online learning of object kinematics from perceptual and contact interaction into the closed-loop policy learning of our approach.

## ACKNOWLEDGEMENT

This research was supported by NSF CAREER Award 1844960 to Konidaris, and by the ONR under the PERISCOPE MURI Contract N00014-17-1-2699. Disclosure: George Konidaris is the Chief Roboticist of Realtime Robotics, a robotics company that produces a specialized motion planning processor.

## REFERENCES

[1] M. P. Deisenroth, G. Neumann, J. Peters *et al.*, "A survey on policy search for robotics," *Foundations and Trends® in Robotics*, vol. 2, no. 1–2, pp. 1–142, 2013.

[2] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robotics and autonomous systems*, vol. 57, no. 5, pp. 469–483, 2009.

[3] S. Levine and V. Koltun, "Guided policy search," in *International Conference on Machine Learning*, 2013, pp. 1–9.

[4] 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," *arXiv preprint arXiv:1709.10087*, 2017.

[5] C.-A. Cheng, X. Yan, N. Wagener, and B. Boots, "Fast Policy Learning through Imitation and Reinforcement," *arXiv e-prints*, p. arXiv:1805.10413, May 2018.

[6] S. Schaal, "Is imitation learning the route to humanoid robots?" *Trends in cognitive sciences*, vol. 3, no. 6, pp. 233–242, 1999.

[7] Y. Pan, C.-A. Cheng, K. Saigol, K. Lee, X. Yan, E. Theodorou, and B. Boots, "Agile autonomous driving using end-to-end deep imitation learning," *arXiv preprint arXiv:1709.07174*, 2017.

[8] T. Tosun, E. Mitchell, B. Eisner, J. Huh, B. Lee, D. Lee, V. Isler, H. S. Seung, and D. Lee, "Pixels to plans: Learning non-prehensile manipulation by imitating a planner," *arXiv preprint arXiv:1904.03260*, 2019.

[9] T. Jurgenson and A. Tamar, "Harnessing reinforcement learning for neural motion planning," *arXiv preprint arXiv:1906.00214*, 2019.

[10] Y. Jiang, F. Yang, S. Zhang, and P. Stone, "Task-motion planning with reinforcement learning for adaptable mobile service robots." in *IROS*, 2019, pp. 7529–7534.

[11] A. J. Ijspeert, J. Nakanishi, and S. Schaal, "Learning attractor landscapes for learning motor primitives," in *Advances in neural information processing systems*, 2003, pp. 1547–1554.

[12] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *Journal of Machine Learning Research*, vol. 17, no. 39, pp. 1–40, 2016. [Online]. Available: http://jmlr.org/papers/v17/15-522.html

[13] C. G. Atkeson and S. Schaal, "Robot learning from demonstration," in *ICML*, vol. 97. Citeseer, 1997, pp. 12–20.

[14] P. Pastor, H. Hoffmann, T. Asfour, and S. Schaal, "Learning and generalization of motor skills by learning from demonstration," in *2009 IEEE International Conference on Robotics and Automation*. IEEE, 2009, pp. 763–768.

[15] J. Ho and S. Ermon, "Generative adversarial imitation learning," in *Advances in neural information processing systems*, 2016, pp. 4565–4573.

[16] S. Ross, G. Gordon, and D. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, 2011, pp. 627–635.

[17] A. Nair, D. Chen, P. Agrawal, P. Isola, P. Abbeel, J. Malik, and S. Levine, "Combining self-supervised learning and imitation for vision-based rope manipulation," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 2146–2153.

[18] W. Sun, A. Venkatraman, G. J. Gordon, B. Boots, and J. A. Bagnell, "Deeply aggrevated: Differentiable imitation learning for sequential prediction," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 3309–3318.

[19] T. Lozano-Perez, "Automatic planning of manipulator transfer movements," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 11, no. 10, pp. 681–698, 1981.

[20] S. M. LaValle, *Planning algorithms*. Cambridge university press, 2006.

[21] B. Abbatematteo, S. Tellex, and G. Konidaris, "Learning to generalize kinematic models to novel objects," *Proceedings of the 3rd Conference on Robot Learning*, 2019.

[22] X. Li, H. Wang, L. Yi, L. Guibas, A. L. Abbott, and S. Song, "Category-level articulated object pose estimation," *arXiv preprint arXiv:1912.11913*, 2019.

[23] S. Schaal and C. G. Atkeson, "Constructive incremental learning from only local information," *Neural computation*, vol. 10, no. 8, pp. 2047–2084, 1998.

[24] E. Coumans *et al.*, "Bullet physics library," *Open source: bulletphysics. org*, vol. 15, no. 49, p. 5, 2013.

[25] S. M. Kakade, "A natural policy gradient," in *Advances in neural information processing systems*, 2002, pp. 1531–1538.

[26] F. Stulp and O. Sigaud, "Path integral policy improvement with covariance matrix adaptation," *arXiv preprint arXiv:1206.4621*, 2012.

[27] A. Campeau-Lecours, H. Lamontagne, S. Latour, P. Fauteux, V. Maheu, F. Boucher, C. Deguire, and L.-J. C. L'Ecuyer, "Kinova modular robot arms for service robotics applications," in *Rapid Automation: Concepts, Methodologies, Tools, and Applications*. IGI Global, 2019, pp. 693–719.

[28] S. Chitta, I. Sucan, and S. Cousins, "Moveit![ros topics]," *IEEE Robotics & Automation Magazine*, vol. 19, no. 1, pp. 18–19, 2012.

[29] S. Karaman, M. R. Walter, A. Perez, E. Frazzoli, and S. Teller, "Anytime motion planning using the rrt," in *2011 IEEE International Conference on Robotics and Automation*. IEEE, 2011, pp. 1478–1483.

[30] A. Kurenkov, A. Mandlekar, R. Martin-Martin, S. Savarese, and A. Garg, "Ac-teach: A bayesian actor-critic method for policy learning with an ensemble of suboptimal teachers," *arXiv preprint arXiv:1909.04121*, 2019.

[31] T. Silver, K. Allen, J. Tenenbaum, and L. 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 *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 6023–6029.

[33] Y. Chebotar, M. Kalakrishnan, A. Yahya, A. Li, S. Schaal, and S. Levine, "Path integral guided policy search," in *2017 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2017, pp. 3381–3388.

[34] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.