ADAPT: Zero-Shot Adaptive Policy Transfer for Stochastic Dynamical Systems

James Harrison*[1], Animesh Garg*[2], Boris Ivanovic[2], Yuke Zhu[2], Silvio Savarese[2], Li Fei-Fei[2], Marco Pavone[3] *denotes equal contribution

Abstract Model-free policy learning has enabled robust performance of complex tasks with relatively simple algorithms. However, this simplicity comes at the cost of requiring an Oracle and arguably very poor sample complexity. This renders such methods unsuitable for physical systems. Variants of model-based methods address this problem through the use of simulators, however, this gives rise to the problem of policy transfer from simulated to the physical system. Model mismatch due to systematic parameter shift and unmodelled dynamics error may cause sub-optimal or unsafe behavior upon direct transfer. We introduce the Adaptive Policy Transfer for Stochastic Dynamics (ADAPT) algorithm that achieves provably safe and robust, dynamically-feasible zero-shot transfer of RL-policies to new domains with dynamics error. ADAPT combines the strengths of offline policy learning in a black-box source simulator with online tube-based MPC to attenuate bounded model mismatch between the source and target dynamics. ADAPT allows online transfer of policy, trained solely in a simulation offline, to a family of unknown targets without fine-tuning. We also formally show that (i) ADAPT guarantees state and control safety through state-action tubes under the assumption of Lipschitz continuity of the divergence in dynamics and, (ii) ADAPT results in a bounded loss of reward accumulation in case of direct transfer with ADAPT as compared to a policy trained only on target. We evaluate ADAPT on 2 continuous, non-holonomic simulated dynamical systems with 4 different disturbance models, and find that ADAPT performs between 50%-300% better on mean reward accrual than direct policy transfer.

James Harrison
Department of Mechanical Engineering, Stanford University, Stanford, CA 94305
e-mail: jh2@stanford.edu

Animesh Garg, Boris Ivanovic, Yuke Zhu, Li Fei-Fei, Silvio Savarese
Department of Computer Science, Stanford University, Stanford, CA 94305
e-mail: {garg,borisi,yukez,feifeili,ssilvio}@cs.stanford.edu

Marco Pavone
Department of Aeronautics and Astronautics, Stanford University, Stanford, CA 94305
e-mail: pavone@stanford.edu

1 Introduction

Deep reinforcement learning (RL) has achieved remarkable advances in sequential decision making in recent years, often outperforming humans on tasks such as Atari games [24]. Deep networks allow representation of analytically intractable dynamics functions [25], thus enabling model-based policy learning for complex dynamics models [4], [26]. However, model-free variants of deep RL are not directly applicable to physical systems because they exhibit poor sample complexity often requiring millions of training examples on an oracle, aka a perfect environment model.

This problem can be assuaged with Model-based RL, i.e. first learn a policy in a source domain (e.g., simulation), with low-cost evaluation and resets, and then transfer it to the target domain (e.g., a real-world robot) either directly [33], [38], [41] or with fine-tuning [1], [31]. On the contrary, the control literature suggests the use of model-predictive control (MPC) for such methods with a simplified model. However, such methods still need an analytic model and can get stuck in local minima if the solution is not initialized properly.

This paper studies the challenge of training in simulation and directly operating in a target environment as the *policy adaptation in direct transfer* problem as reviewed in [37]. The direct transfer involves training a policy on a system possessing different dynamics than the target system, and evaluating performance as the average initial return in target domain *without* training in the target domain. In the machine learning and reinforcement learning literature, this is also known as *zero-shot* transfer. This problem is challenging for robotic systems since simplified simulated models may not always accurately capture all relevant dynamics phenomena, such as friction, structural compliance, turbulence and so on, as well as parametric uncertainty in the model. In spite of the renewed focus on this problem, few studies in deep policy adaptation offer insightful analysis and guarantees regarding feasibility, safety, and robustness in policy transfer.

We introduce a new algorithm Adaptive Policy Transfer for Stochastic Dynamics (ADAPT) that achieves provably safe and robust, dynamically-feasible zero-shot direct transfer of RL-policies to new domains with dynamics error. The key insight here is to leverage the global optimality of learned policy with local stabilization from MPC based methods to enable dynamic feasibility, thereby building on strengths of two different methods. In the offline stage, ADAPT first computes offline a nominal trajectory (without disturbance) by executing the learned policy on the simulator dynamics. Then in the online stage, ADAPT, fittingly, adapts the nominal trajectory to the target dynamics with an auxiliary MPC controller.

Statement of Contributions

1. ADAPT allows online transfer of policy, trained solely in a simulation offline, to a family of unknown targets without fine-tuning.
2. We also formally show that (i) ADAPT guarantees state and control safety through state-action tubes under the assumption of Lipschitz continuity of the divergence in dynamics and, (ii) ADAPT results in a bounded loss of reward accumulation in case of direct transfer with ADAPT as compared to a policy trained only on target.

3. We evaluate ADAPT on two continuous, non-holonomic simulated dynamical systems with four different disturbance models, and find that ADAPT performs between 50%-300% better on mean reward accrual than direct policy transfer as compared to mean reward.

Organization This paper is structured as follows. In Section 2 we review related work in robust control, robust reinforcement learning, and transfer learning. In Section 3 we formally state the policy transfer problem. In Section 4 we present ADAPT and discuss design parameters. In Section 5 we prove the accrued reward for ADAPT is lower bounded. In Section 6 we present experimental results on a simulated car environment and a two-link robotic manipulator. Finally, in Sections 7 and 8 we discuss the performance of ADAPT with robust policy learning methods, as well as draw conclusions and discuss future directions.

2 Related Work and Background

A plethora of work in both the learning and control theory has addressed the problem of varying system dynamics, especially in the context of safe policy transfer and robust control.

Transfer in reinforcement learning The problem of high sample complexity in reinforcement learning has generated considerable interest in policy transfer. Taylor et al. provide an excellent treatise on the transfer learning problem [37]. A series of approaches focused on reducing the number of rollouts performed on a physical robot, by alternating between policy improvement in simulation and physical rollouts [1], [20]. In those work, a time-dependent term is added to the dynamics after each physical rollout to account for unmodeled error. This approach, however, does not address robustness in the initial transfer, and the system could sustain or cause damage before the online learning model converges.

The EPOPT algorithm [31] randomly samples dynamics parameters from a Gaussian distribution prior to each training run, and optimizes the reward for the worst-performing ϵ-fraction of dynamics parameters. However, it is not clear how robust it is against disturbances not explicitly experienced in training. This approach is conceptually similar to that in [27], in which more traditional trajectory optimization methods are used with an ensemble of models to increase robustness.Similarly, [21] use adversarial samples instead of random samples as hard negatives for rbust policy training. In [30], the authors add adversarial disturbances during training. Christiano et al. [5] approach the transfer problem by training an inverse dynamics model on the target system and generating a nominal trajectory of states. The inverse dynamics model then generates actions to connect these states. However, there are no guarantees that an action exists in the target dynamics to connect two learned adjacent states. Moreover, this requires training on the target environment; in this work we consider zero-shot learning where this is not possible. Recently, the problem of transfer has been addressed in part by rapid test adaptation [9], [32]. These approaches have focused on training modular networks that have both "task-specific" and "robot-specific" modules. This then allows the task-specific module to be effi-

ciently swapped out and retrained. However, it is unclear how learned model error affects these methods.

In this work we aim to perform zero-shot policy transfer, and thus efficient model-based approaches are not directly applicable. However, our approach uses an auxiliary control scheme that leverages model learning for an approximate dynamics model. When online learning is possible, sample-efficient model-based reinforcement learning approaches can dramatically improve sample complexity, largely by leverage tools from planning and optimal control [18]. However, these models require an accurate estimate of the true system dynamics in order to learn an effective policy. A variety of model classes have been used to represent system dynamics, such as neural networks [12], [14], Gaussian processes [7], [8], and local linear models [13], [20]. Several works start with a dynamical model in simulation, and perform dynamics adaptation on the real system [1], [5]. However, these approaches generally require the model to be fit to experimental data, which must be acquired under similar circumstances to the operational conditions.

Robust control Trajectory optimization methods have been widely used for robotic control [36]. Among these optimization methods, model predictive control (MPC) is a class of online methods that perform trajectory optimization in a receding-horizon fashion [29]. This receding-horizon approach, in which a finite-horizon, open-loop trajectory optimization problem is continuously re-solved, results in an online control algorithm that is robust to disturbances. Several works have attempted to combine trajectory optimization methods with dynamics learning [23] and policy learning [15]. In this work, we develop an auxiliary robust MPC-based controller to guarantee robustness and performance for learned policies. Our method combines the strengths of deep policy networks [34] and tube-based MPC [22] to offer a well-performing controller with safety guarantees. Our auxiliary controller is based on an iterative solution of the nonlinear MPC problem. Approaches similar to the auxiliary controller we present here have been developed to allow robotics systems to generate trajectories of several seconds in a couple of milliseconds [29].

3 Problem Setup and Preliminaries

Consider a finite-horizon Markov Decision Process ($\mathcal{M}$) defined as a tuple $\mathcal{M} : \langle \mathcal{S}, \mathcal{A}, \mathcal{T}, R, T \rangle$. Here $\mathcal{S}$ and $\mathcal{A}$ represent continuous, bounded state and action spaces for the agent, $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function that maps a state-action tuple to a scalar, and T is the problem horizon. Finally, $\mathcal{T}$ is a transition function that captures the state transition dynamics in the environment and is a distribution over states conditioned on the previous state and action. The goal is to find a policy $\pi : \mathcal{S} \to \mathcal{A}$ that maximizes the expected cumulative reward $\eta(\pi)$ over the choice of policy:

$$\pi^*(s) = \underset{\pi(s)}{\operatorname{argmax}}\, \eta(\pi) = \underset{\pi(s)}{\operatorname{argmax}}\, \mathbb{E}\left[\sum_{t=0}^{T} R(s_t, a_t)\right], \tag{1}$$

We assume that a black-box simulator model that emulates the true target is available, however, we make a weak assumption on its correctness, specifically, bounded uncertainty. We assume the simulator (denoted $\mathcal{M}_S$) has twice continuously-

differentiable dynamics $s_{t+1} = f(s_t, a_t)$. Then, let the dynamics of the target environment (denoted $\mathcal{M}_T$) be denoted $s_{t+1} = g(s_t, a_t)$. For all state action pairs $(s,a) \in \mathcal{S} \times \mathcal{A}$, the error between sets of dynamics, $w = \|f(s,a) - g(s,a)\|$ lies in a compact, convex set $\mathcal{W}$ that contains the origin. Based on the above definitions, we can now state the problem we aim to solve.

Problem Statement Given the simulator dynamics and the problem defined by the MDP $\mathcal{M}_S$, we wish to learn a policy to maximize the reward accrued during operation in the target system. Formally, if we write the realization of the disturbance at time t as w_t, we wish to solve the problem:

$$\begin{aligned} &\max_{\mathbf{a}} \mathbb{E}\left[\sum_{t=0}^{T} R(s_t, a_t)\right] \\ &\text{s.t.} \quad s_{t+1} = f(s_t, a_t) + w_t, \text{ and } s_t \in \mathcal{S},\, a_t \in \mathcal{A} \quad \forall t \in [0, T], \end{aligned} \tag{2}$$

while only having access to the simulator, $\mathcal{M}_{\mathcal{S}}$, for training and no training on the target, $\mathcal{M}_{\mathcal{T}}$.

4 ADAPT: Adaptive Policy Transfer for Stochastic Dynamics

In this section we present the ADAPT algorithm for zero-shot transfer. A high level view of the algorithm is presented in Algorithm 1. First, we assume that a policy is trained in simulation. Our approach is to first compute a nominal trajectory (without disturbance) by continuously executing the learned policy on the simulator dynamics. Then, when transferred to the target environment, we use an auxiliary model predictive control (MPC) controller to stabilize around this nominal trajectory.

Policy Training We use model-free policy optimization on the black-box simulated model. Our theoretical guarantees rely on the auxiliary controller avoiding saturation. Therefore, if a policy operates near the limits of its control authority and thus saturates when used with the auxiliary controller, this policy is trained using restricted state and action spaces $\mathcal{S}' \subseteq \mathcal{S}$, $\mathcal{A}' \subseteq \mathcal{A}$. We let $\mathcal{M}'$ denote an MDP with restricted state and action spaces. This follows the approach of [22], where it is used to prevent auxiliary controller saturation. Intuitively, restricting the state and action space ensures any nominal trajectory in those spaces can be stabilized by the auxiliary controller. Therefore, if saturation is rare, restricting these sets is unnecessary.

ADAPT is invariant to the choice of policy optimization method. During online operation, a nominal trajectory $\tau = \{(\bar{s}_t, \bar{a}_t)\}_{t=0}^{T}$ is generated by rolling out the policy on the simulator dynamics, $\mathcal{M}_{\mathcal{S}}$. The auxiliary controller then tracks this trajectory in the target environment.

Approximate Dynamics Model Because the model of the simulator is treated as a black-box, it is impractical to use for the auxiliary controller in an optimal control framework. As such, we rely on an approximate model of the dynamics, separate from the simulator dynamics f, which we refer to as $\hat{f}$. The specific representation of the model (e.g. linear model, feedforward neural network, etc.) depends on both the accuracy required as well as the method used to solve the auxiliary control problem. This model may be either learned from the simulator, or based on prior knowledge.

Algorithm 1 Adaptive Policy Transfer for Stochastic Dynamics (ADAPT)

Input: Source Env: $\mathcal{M}_S$, Target Env: $\mathcal{M}_T$, Initial State: s_0

Offline:

1: $\mathcal{A}', \mathcal{S}' \leftarrow \texttt{bound\_set}(\mathcal{A}, \mathcal{S})$ // Calculate constrained state & action space
2: $\pi \leftarrow \texttt{policy\_opt}(\mathcal{M}'_S)$ // Train a policy for $\mathcal{M}'_S$ using constrained $\mathcal{S}', \mathcal{A}'$
3: $\hat{f} \leftarrow \texttt{fit\_dynamics}(\mathcal{M}_S)$ // Fit Dynamics for $\mathcal{M}_S$

Online:

4: $\tau \leftarrow \texttt{rollout}(s_0, \pi, \mathcal{M}_S, T)$ // Roll out π on $\mathcal{M}_S$ to get nominal trajectory
5: $s \leftarrow s_0$
6: **for** $t \in [0, T]$ **do**
7: $\quad a \leftarrow \texttt{aux\_MPC}(s, \tau, \hat{f}, \tau, N)$ // NMPC with iterative linearization
8: $\quad s \leftarrow f(s, a) + w$ // Rollout the first step of action seq. on $\mathcal{M}_T$
9: **end for**

A substantial body of literature exists on dynamics model learning from black-box systems [25]. Alternatively, this model may be based on external knowledge, either from learning a dynamics model in advance from the target system or from, for example, a physical model of the system.

Auxiliary MPC Controller Our auxiliary nonlinear MPC controller is based on that of [22]. Specifically, we write the auxiliary control problem:

$$\begin{aligned} \min_{\mathbf{a}} \quad & \sum_{k=t}^{t+N} (s_k - \bar{s}_k)^T Q (s_k - \bar{s}_k) + (a_k - \bar{a}_k)^T R (a_k - \bar{a}_k) \\ \text{s.t.} \quad & s_{k+1} = \hat{f}(s_k, a_k), \text{ and } s_k \in \mathcal{S},\, a_k \in \mathcal{A} \quad \forall k \in [t, t+N], \end{aligned} \tag{3}$$

where N is the MPC horizon, Q and R are cost matrices for the state deviation and control deviation respectively, and $\hat{f}$ is the approximate dynamics model. In some cases, this problem is convex, but generally it may not be. In our experiments, this optimization problem is solved with iterative relinearization based on [39]. However, whereas they iteratively linearize the nonlinear optimal control problem and solve an LQR problem over the full horizon of the problem, we explicitly solve the problem over the MPC horizon. We do not consider terminal state costs or constraints. This formulation of the auxiliary controller by [22] allows us to guarantee, under our assumptions, that our true state stays in a tube around the nominal trajectory, where the tube is defined by level sets of the value function (the details of this are addressed in Section 6).

The solution to the MPC problem is iterative. First, we linearize around the nominal trajectory τ. We introduce the notation $\{(\hat{s}_k, \hat{a}_k)\}_{k=t}^{k=t+N}$, which is the solution for the last iteration. These are initialized as $\hat{s}_t \leftarrow \bar{s}_t$ and $\hat{a}_t \leftarrow \bar{a}_t$. Then, we introduce the deviations from this solution as

$$\delta s_t = s_t - \hat{s}_t, \qquad \delta a_t = a_t - \hat{a}_t. \tag{4}$$

Then, taking the linearization of our dynamics

$$A_t = \left.\frac{\partial \hat{f}}{\partial s_t}\right|_{s_t = \hat{s}_t, a_t = \hat{a}_t} \qquad B_t = \left.\frac{\partial \hat{f}}{\partial a_t}\right|_{s_t = \hat{s}_t, a_t = \hat{a}_t}, \tag{5}$$

we can rewrite the MPC problem as:

$$\begin{aligned} \min_{\delta \mathbf{a}} \quad & \sum_{k=t}^{t+N} (\delta s_k + \hat{s}_k - \bar{s}_k)^T Q (\delta s_k + \hat{s}_k - \bar{s}_k) + (\delta a_k + \hat{a}_k - \bar{a}_k)^T R (\delta a_k + \hat{a}_k - \bar{a}_k) \\ \text{s.t.} \quad & \delta s_{k+1} = A_k \delta s_k + B_k \delta a_k, \text{ and } \delta s_k + \hat{s}_k \in \mathcal{S}, \, \delta a_k + \hat{a}_k \in \mathcal{A}, \quad \forall k \in [t, t+N]. \end{aligned} \tag{6}$$

Note that the optimization is over δa_k's. Once this problem is solved, we use the update rule $\hat{s}_t \leftarrow \hat{s}_t + \delta s_t$, $\hat{a}_t \leftarrow \hat{a}_t + \delta a_t$. The dynamics are then relinearized, and this is iterated until convergence. Because we use iterative linearization to solve the nonlinear program, it is necessary to choose a dynamics representation $\hat{f}$ that is efficiently linearizable. In our experiments, we use an analytical nonlinear dynamics representation for which the linearization can be computed analytically (see [40] for details), as well as fit a time-varying linear model. Choices such as, e.g., a Gaussian process representation, may be expensive to linearize.

5 ADAPT: Analysis

The following section develops the main theoretical analysis of this study. We will first show that ADAPT zero-shot transfers to a new MDP with bounded deviation from nominal state and control output from the learned trajectory. Then we will show that the ADAPT trajectory output after online adaptation results in bounded loss in cumulative reward in target as compared to training from scratch.

5.1 Safety Analysis in ADAPT

Using the notation from Eq (6), let us denote the optimal cost as $C^*(s)$. It is a measure of the distance between the state of the system and the reference trajectory. Now, suppose $C^*(s)$ has a value *zero* on the reference trajectory, any level set of the MPC-cost defines a neighborhood of the reference trajectory, or a tube around reference. We can define this set as the level set $L_d(t) \triangleq \{s | C^*(s) \leq d\}$, $\forall d \in (0, c)$, where c is some constant. This in turn defines state tube $\mathbb{T}_s \triangleq \{L_d(s;t) | 1 \leq t \leq N\}$ and action tube $\mathbb{T}_a \triangleq \{\pi(L_d(\hat{s};t)) | 1 \leq t \leq N\}$.

Further we assume, (i) Source dynamics $f(\cdot)$ is twice continuously differentiable, (ii) The error w between target and source dynamics lies in a compact convex set $\mathcal{W}$ containing the origin, (iii) Q and R are positive definite, then we can state the following about (iv) there exists a bounded state $\mathcal{S}' \subseteq \mathcal{S}$ and action $\mathcal{A}' \subseteq \mathcal{A}$ set, such that the optimal level set is constrained $L_{d^*}(s;t) \subset \mathcal{S}$.

Theorem 1 (Safety in ADAPT). *Under the assumptions above, for each initial state s, that is stabilizable, every state and action trajectory under the target dynamics $\mathcal{T}_T$ with the initial state $s(0) = s_0$ will lie in the state $\mathbb{T}_s$ and action $\mathbb{T}_a$ tubes respectively. The state and action trajectories also satisfy boundary constraints at each intermediate time step as well.*

The observation above is a result of the Feedback-MPC, stating that $\mathbb{T}_s$ and $\mathbb{T}_a$ are invariant for the target dynamics system in the sense that any state (and action) trajectory starting inside the tube stays inside the tube under the ADAPT controller. This result shows that the auxiliary controller maintains the state of the target system in a neighbourhood of the reference trajectory obtained from the source dynamics.

The neighbourhood is a level-set of the cost function for the ancillary control problem. We refer the reader to an excellent treatise on tube-based feedback MPC by Mayne et al.[22] and Singh et al. [35] for analysis of bounded state tubes around nominal policy output under bounded disturbances on nominal model.

5.2 Robustness Analysis in ADAPT

Proposition 1. *Let $\mathcal{M}$ be a discounted MDP and V be an arbitrary value function. The value function of the greedy policy based on V is denoted by V^{π}. Then,*

$$\|V^* - V^{\pi}\|_{\infty} \leq \frac{2\gamma}{1-\gamma}\|V^* - V\|_{\infty},$$

where $\|\cdot\|_{\infty}$ denotes the max-norm over all states. Moreover if $\exists\, \varepsilon > 0$, s.t. if $\|V^* - V\|_{\infty} < \varepsilon$, then $V^{\pi} = V^*$.

The above result states that if V is "close" to optimal V^* then the greedy policy with one stage look-ahead based on V will also be "close" to the optimal policy. We refer the reader to Bertsekas and Tsitsiklis [2] for the proof. Intuitively, this also says that if a *good* approximation of the optimal value function is available, then a *good* policy can be easily obtained, for instance a greedy policy to the approx. value function. Hence we should focus effort on calculating a good approx. of the value function in the target MDP $\mathcal{M}_T$ without a target oracle.

Now we will investigate what happens if the model is inaccurate or the environment differs, as is in our problem setup. This question has been inquired by a number of past works, in particular by [6], [11], [16], [17], [28]. Studies such as [6], [16], [28] studied this in context of robust MDPs and [17] proposed a simulation lemma to give a polynomial time bound to achieve near optimal return in approximation errors in MDP transition or cost functions. However, none of these methods make the connection between a practical method to calculate a source policy, and perform a robust transfer, as performed in this paper.

We want the approximate value function of a fixed policy, obtained in source MDP, with respect to the target MDP. To get this as a first step, we show that the the optimal value function varies Lipschitz continuously with the transition dynamics.

Lemma 1. *If the two MDPs differ only in their transition function, then the difference in the corresponding value functions V_S^* and V_T^* is bounded by:*

$$\|V_S^* - V_T^*\|_{\infty} \leq \frac{n\gamma\|r\|_{\infty}}{(1-\gamma)^2}\|\mathcal{T}_S - \mathcal{T}_T\|_{\infty},$$

where n is state-space size and γ is discount factor

However, we note that the size of the state-space (n) can affect the Lipschitz constant adversely. As suggested in [6], instead of using the max error in a single transition, we can consider a state-action pair where dynamics differs the most, in aggregate.

Proposition 2. *With the assumptions in lemma 1, if we consider an induced matrix norm between transition dynamics, $\|\mathcal{T}(\omega_S)\|_1 = max_{s,a}(\sum_{s'}|\mathcal{T}(s,a;\omega_S)|)$, then we have an improved upper bound on the divergence of the value functions:*

$$\|V_S^* - V_T^*\|_{\infty} \leq \frac{\gamma\|r\|_{\infty}}{(1-\gamma)^2}\|\mathcal{T}_S - \mathcal{T}_T\|_1$$

The induced matrix norm in case of additive noise reduces to the max norm of the noise, i.e. $\|\mathcal{T}_S - \mathcal{T}_T\|_1 = \|w\|_\infty$. This result in essence captures the fact that optimal value functions vary L.C. with dynamics with a Lipschitz constant $\frac{\gamma}{(1-\gamma)^2}$ times the max reward. We omit the proof here for brevity. Please refer to Kalmar et al. [16].

We have shown that the change in optimal value functions is bounded. Now, we will show that the value function from the optimal policy π_S^* obtained from source MDP $\mathcal{M}_S$ and applied in target $\mathcal{M}_T$ with auxiliary feedback.

Lemma 2. *The difference in optimal value function of the MDP $\mathcal{M}_S$ with corresponding value function for* ADAPT *policy on target MDP $\mathcal{M}_T$ varies Lipschitz continuously, such that,*

$$\|V_S^* - V_T(\hat{\pi}_S^*)\|_\infty \leq \frac{\gamma \|r\|_\infty}{(1-\gamma)^2} \|\mathcal{T}_S - \mathcal{T}_T\|_1,$$

Proof. Let, a policy in the source MDP $\mathcal{M}_S$ be π_S and corresponding value function be $V_S(\pi_S)$. The corresponding value function of π_S on source MDP $\mathcal{M}_T$ is $V_T(\pi_S)$. Further, using an auxiliary controller on π_S results in a policy $\hat{\pi}_S$, and results in a value function $V_T(\hat{\pi}_S)$. Similarly, for optimal policy π_S^* results in $V_S(\pi_S^*)$ and $V_T(\hat{\pi}_S^*)$. Using proposition 2 for policy $\hat{\pi}_S^*$, instead of the optimal greedy policy,

$$\|V_S(\pi_S^*) - V_T(\hat{\pi}_S^*)\|_\infty \leq \frac{\gamma \|r\|_\infty}{(1-\gamma)^2} \|\mathcal{T}_S - \mathcal{T}_T\|_1.$$

We note that when $\hat{\pi}_S^*$ is applied to MDP $\mathcal{M}_S$, the the output of auxiliary controller is all zero since there is no dynamics noise. Hence $\hat{\pi}_S^* = \pi_S^*$, under dynamics $\mathcal{T}_S$, resulting in $V_S(\hat{\pi}_S^*) = V_S(\pi_S^*)$. Further noting, that since π_S^* was obtained using policy optimization until convergence, hence $V_S(\pi_S^*)$ is indeed V_S^*. Hence the result follows.

Finally we show that the difference of $V_T(\hat{\pi}_S^*)$ from the optimal value function of target V_T^* is bounded and hence $V_T(\hat{\pi}_S^*)$ is a good approximation of V_T^*.

Theorem 2 (Bounded Performance with ADAPT). *The main theorem provides a bound on robustness of the* ADAPT *policy when applied to target MDP $\mathcal{M}_T$.*

$$\|V_T^* - V_T^{\pi_S}\|_\infty \leq \frac{2\gamma \|r\|_\infty}{(1-\gamma)^2} \|\mathcal{T}_S - \mathcal{T}_T\|_1,$$

Proof. Combining the results from preposition 2 and lemma 2, and applying triangle inequality yields the result.

In addition to providing a bound on value functions, a corollary of Theorem 2 is that a policy π_S^* trained on source MDP $\mathcal{M}_S$, and used on target MDP $\mathcal{M}_T$ with ADAPT as policy $\hat{\pi}_S^*$ results in a bounded regret in T-step reward as compared to a policy π_T^* trained on target MDP $\mathcal{M}_T$. This results from combining Theorem 2 and the simulation lemma from [17].

So summarily, we have stated that a policy completely trained in a simulated environment and transferred to a unknown target environment without fine-tuning can result in bounded state deviation, as shown in Theorem 1, as well as robust performance, as shown in Theorem 2, as compared to training from scratch in target.

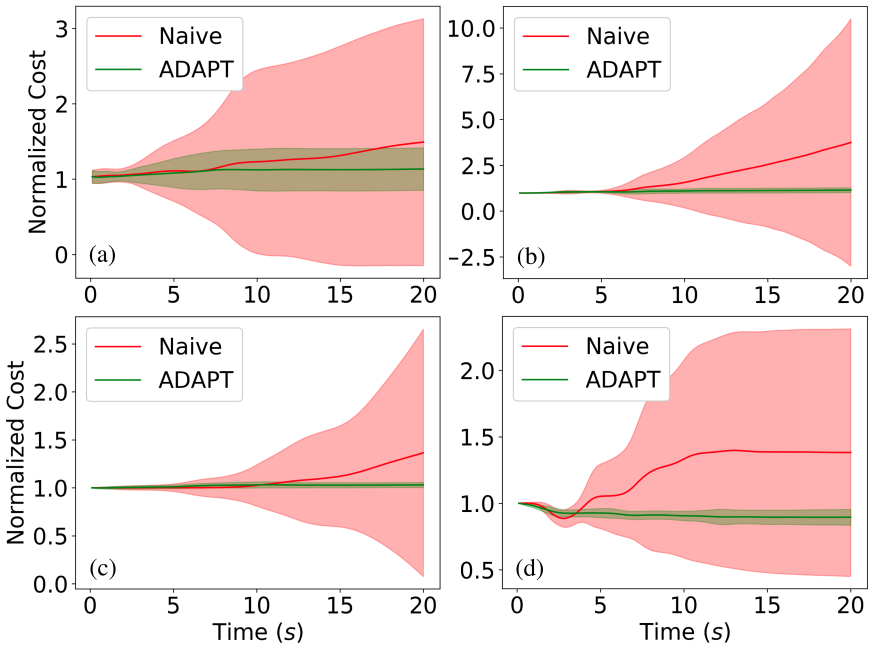


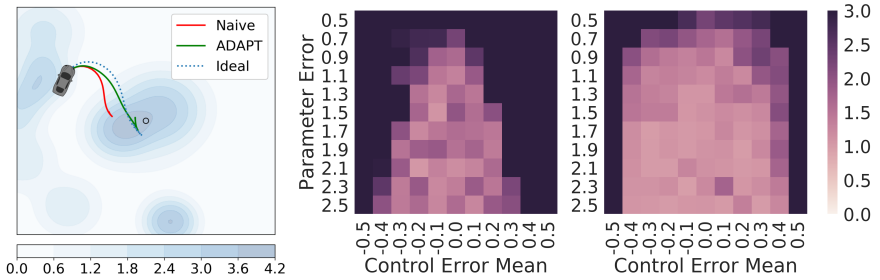
Fig. 1: Mean cumulative cost over the length of an episode for 50 episodes on the kinematic car environment. The confidence intervals are standard error. The costs are normalized to the cost of the naive policy being rolled out on the simulated environment from the same initial state, to allow more direct comparison across episodes. The *naive* rollout is the nominal policy executed on the target environment. The disturbances tested are **a)** a hill landscape, **b)** additive control error, **c)** process noise, and **d)** dynamics parameter error.

6 Experimental Evaluation

We implemented ADAPT on a nonlinear, non-holonomic kinematic car model with a 5-dimensional state space as well as on the `Reacher` environment in OpenAI's Gym [3]. We train policies using Trust Region Policy Optimization (TRPO) [34]. The policy is parameterized as a neural network with two hidden layers, each with 64 units and ReLU nonlinearities. In all of our experiments, we report *normalized cost*. This is the cost (negative reward) realized by a trial in the target environment, divided by the cost of the nominal policy rolled out on the simulated environment from the same initial state. This allows more direct comparison between episodes for environments with stochastic initial states. We generally compare the *naive* trial, which is the nominal policy rolled out on the target environment (e.g., standard transfer with no adaptation) to ADAPT.

6.1 Environment I: 5-D Car

We implemented ADAPT on a nonlinear, nonholonomic 5-dimensional kinematic car model that has been used previously in the motion planning literature [40]. Specifically, the car has state $s = [x, y, \theta, v, \kappa]^T$, where x and y denote coordinates in the



(a) (b) (c)

Fig. 2: **a)** The car environment with the paths for the ideal case (nominal policy on simulated environment), the naive case (nominal policy on the target environment), and the ADAPT case (ADAPT on the target environment). The contour plot shows the height of the added hills. Figures (b) and (c) show the normalized cost for varying disturbances due to additive control error and dynamics parameter error for **b)** the naive case and **c)** ADAPT (lower is better). In addition to the listed disturbances, disturbances due to hills are also added for all trials. Each grid cell is the mean of 50 trials.

plane, θ denotes heading angle, v denotes speed, and κ denotes trajectory curvature. The system has dynamics $\dot{s} = [v\cos\theta, v\sin\theta, v\kappa, a_v, a_\kappa]$, where $a_v \in [-2,2]$ and $a_\kappa \in [-0.5,0.5]$ are the controlled acceleration and curvature derivative. The policy is trained to minimize the quadratic cost $L(\boldsymbol{s},\boldsymbol{a}) = \sum_{t=0}^{T} \ell(s_t,a_t)$, where $\ell(s_t,a_t) = x_t^2 + y_t^2 + a_{v,t}^2 + a_{\kappa,t}^2$, which results in policies that drive to the origin. In each trial, the vehicle is initialized in a random state, with position $x,y \in [-5,5]$, with random heading and zero velocity and curvature.

Our auxiliary controller used an MPC horizon of 2 seconds (20 timesteps). Our state deviation penalty matrix, Q, has value 1 along the diagonal for the position terms, and zero elsewhere. Thus, the MPC controller penalizes only deviation in position. The matrix R had small terms (10^{-3}) along the diagonal to slightly penalize control deviations. In practice, this mostly acts as a small regularizing term to prevent large oscillatory control inputs by the auxiliary controller. The behavior of the auxiliary controller is dependent on the matrices Q and R, but in practice good performance may be achieved across environments with fixed values. Because of the relatively high quadratic penalty on control in policy training, the nominal policy rarely approaches the control limits. Thus, we can set $\mathcal{A}' = \mathcal{A}$, and we set $\mathcal{S}' = \mathcal{S}$. For our dynamics model, we use the linearization reported in [40].

6.2 Disturbance Models

We investigate four disturbance types:

1. Environmental Uncertainty: We add randomly-generated hills to the target environment such that the car experiences accelerations due to gravity. This noise is therefore state-dependent. Figure 2a shows a randomly generated landscape. We randomly sample 20 hills in the workspace, each of which is circular and has varying radius and height. The vehicle experiences an additive longitudinal acceleration proportional to the landscape slope at its current location, and no lateral acceleration.

2. Control noise: Nonzero-mean additive control error drawn from a uniform distribution.
3. Process noise: Additive, zero-mean noise added to the state. Disturbances are drawn from a uniform distribution.
4. Dynamics parameter error: We add a scaling factor γ to the control of $\dot{\kappa}$, such that $\dot{\kappa} = \gamma a_\kappa$.

For the last three, the noise terms were drawn i.i.d. from a uniform distribution at each time t. These disturbances were investigated both independently (Figure 1) and simultaneously (Figure 2). Figure 1 shows the normalized cost of the naive transfer and ADAPT for each of the four disturbances individually.

In our experiments, ADAPT substantially outperforms naive transfer, achieving normalized costs 1.5-5x smaller. Additionally, the variance of the naive transfer is considerably higher, whereas the realized cost for ADAPT is clustered relatively tightly around one (e.g., approximately equal cost to the ideal case). In Figure 1d, the normalized cost of ADAPT is actually below one, implying that the transferred policy performs better than the ideal policy. In fact, this is because the dynamics parameter error in this trial results in oversteer, and so the agent accumulates less cost to turn to face the goal than in the nominal environment. Thus, pointing toward the goal is more "cost-efficient" in the target environment. The performance of direct transfer and ADAPT with varying parameter error may be seen in Figure 2b and Figure 2c. In Figure 2a, a case is presented where the direct policy transfer fails to make it up a hill, whereas the ADAPT policy tracks the nominal trajectory well.

6.3 ADAPT *with Robust Offline Policy*

Whereas ADAPT's approach to policy transfer relies primarily on stabilization in the target environment, recent work has focused on training robust policies in the source domain, and then performing direct transfer. In the EPOPT policy training framework [31], an agent is trained over a family of MDPs in which model parameters are drawn from distributions before each training rollout. Then, a Conditional Value-at-Risk (CVaR) objective function is optimized as opposed to an expectation over all training runs. We apply ADAPT on top of an EPOPT-1 policy (equivalent to optimizing expected reward, with model parameters varying), and find that for disturbances explicitly varied during training, the performance of EPOPT-only transfer and ADAPT are comparable. We add parameters γ_i to the state derivative as follows: $\dot{s} = [\gamma_1 v \cos \gamma_2 \theta, \gamma_1 v \sin \gamma_2 \theta, \gamma_1 v \gamma_3 \kappa, \gamma_4 a_v, \gamma_5 a_\kappa]$. Each of these γ_i are drawn from Gaussian distributions before each training run, and are fixed during the training run. Although some of these parameters do not have a physical interpretation, the resulting policies are still robust to both parametric error, as well as process noise. In these experiments, an MPC horizon of 1 second was used (10 timesteps). The matrices Q and R were set as in Section 6.1.

In Figure 3, the comparison between the direct transfer of EPOPT policies and ADAPT policies is presented. We can see that, for disturbances that are explicitly considered in training (specifically, model parameter error), naive transfer performs slightly better, albeit with higher variance. For other disturbances, like the addition

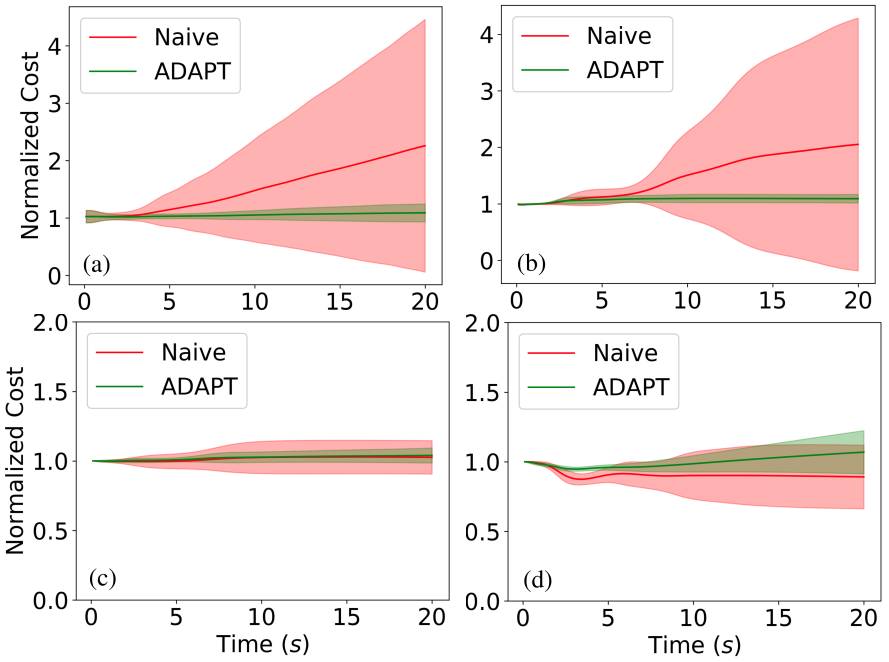


Fig. 3: Mean cumulative cost over the length of an episode for 50 episodes on the 5-D car environment, using an EPOPT-1 robust policy. The confidence intervals are standard error. The disturbances tested are **a)** a hill landscape, **b)** additive control error, **c)** process noise, and **d)** dynamics parameter error. The details of each noise source is presented in the supplementary materials.

of hills or control noise, ADAPT significantly outperforms the directly-transferred policy. Indeed, while the performance of the ADAPT policy is comparable to direct transfer for disturbances directly considered in training, unmodelled disturbances are handled substantially better by ADAPT. Thus, to extract the best performance, we recommend applying the two approaches in tandem.

6.4 Environment II: 2-Link Planar Robot Arm

We next evaluate the performance of ADAPT on the `Reacher` environment of Gym [3]. This environment is a two link robotic arm that receives reward for proximity to a goal in the workspace, and is penalized for control effort. The state is a vector of the sin and cos of the joint angles, as well as joint angular velocities, the goal position, and the distance from the arm end-effector to the goal. In our tests, we fix one goal location and one starting state for all tests to more directly compare between trials. As such, the variance in normalized cost in experiments is much smaller than in the car experiments. For these experiments, the same noise models were used as in the previous section, with the exception of the "hills" disturbance.

As an approximate dynamics model used for the auxiliary controller, we use the time-varying linear dynamics from [19]. This model is fit from rollouts in simulation. Since this model is linear, the MPC problem is convex, and the iterative MPC converges in one iteration. These dynamics are only valid in a local region, and thus

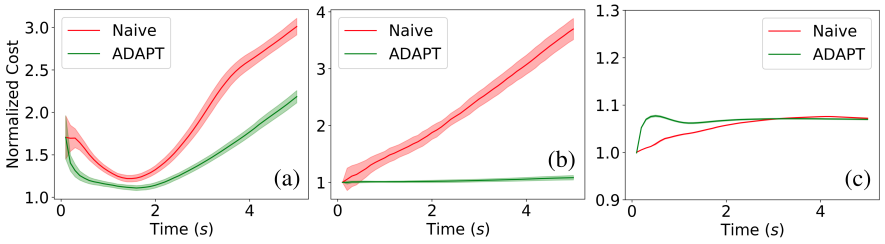


Fig. 4: Mean cumulative cost over the length of an episode for 50 episodes on the reacher environment. The confidence intervals are standard error. The costs are normalized to the cost of the naive policy being rolled out on the simulated environment from the same initial state, to allow more direct comparison across episodes. The *naive* rollout is the nominal policy executed on the target environment. The disturbances tested are **a)** additive control error, **b)** process noise, and **c)** dynamics parameter error.

must be fit for each desired policy rollout in the target environment. However, since the model is fit from simulation data, it is generated quickly and inexpensively.

The results for normalized cost comparisons between naive transfer and ADAPT are presented in Figure 4. We note that ADAPT achieves significantly lower cost for additive control error and process noise, but achieves comparable cost for parameter error. The parameter varied in these experiments was the mass of the links of the arm. The effect of this change is to increase the inertia of the manipulator as a whole. In fact, this can be seen in the Figure 4c. While the cost of the naive transfer increases slowly, the cost of the ADAPT trials spikes at approximately time $t = 0.25$. As ADAPT is tracking the nominal trajectory, it increases the torque applied, thus suffering a penalty for the increased control action, but resulting in better tracking of the nominal trajectory.

A similar effect can be observed in Figure 4a. The added control error actually drives the manipulator toward the goal, resulting in the dip in the normalized cost for both trajectories. However, the naive policy overshoots the goal substantially, and thus accrues substantially higher normalized cost than the ADAPT experiments.

7 Conclusion and Outlook

We have presented the ADAPT algorithm for robust transfer of learned policies to target environments with unmodeled disturbances or model parameters. We have also provided guarantees on the lower bounds of the accrued reward in the target environment for a policy transferred with ADAPT. Our results were demonstrated on two different environments with four disturbance models investigated. We additionally discuss usage of robust policies with ADAPT. The results presented demonstrate that this method improves performance on unmodeled disturbances by 50-300%.

In this work, we construct our analysis on the Lipschitz continuity of the dynamics. Indeed, the smoothness of the deviation in dynamics is fundamental to the guarantees we establish. An immediate avenue of future investigation is, therefore, expanding the work presented here to environments with discrete and discontinuous dynamics such as contact. Recently, Farshidian *et al.* [10] have extended an iteratively linearized nonlinear MPC, similar to ours, to switching linear systems, which may have potential

as a foundation on which to develop a capable contact formulation of ADAPT. Additionally, recent work has developed robust, receding horizon tube controllers that allow the establishment of explicit tubes in the state space [35]. This approach has the potential to establish explicit safety constraints for operation in cluttered environments. Finally, these methods will also be evaluated on a physical systems.

References

[1] P. Abbeel, M. Quigley, and A. Y. Ng, "Using inaccurate models in reinforcement learning", in *Proceedings of the 23rd international conference on Machine learning*, ACM, 2006.

[2] D. P. Bertsekas and J. N. Tsitsiklis, "Neuro-dynamic programming", 1996.

[3] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym", *ArXiv preprint arXiv:1606.01540*, 2016.

[4] Y. Chebotar, K. Hausman, M. Zhang, G. S. Sukhatme, S. Schaal, and S. Levine, "Combining model-based and model-free updates for trajectory-centric reinforcement learning", *CoRR*, vol. abs/1703.03078, 2017.

[5] P. Christiano, Z. Shah, I. Mordatch, J. Schneider, T. Blackwell, J. Tobin, P. Abbeel, and W. Zaremba, "Transfer from simulation to real world through learning deep inverse dynamics model", *ArXiv preprint arXiv:1610.03518*, 2016.

[6] B. C. Csáji and L. Monostori, "Value function based reinforcement learning in changing markovian environments", *Journal of Machine Learning Research*, 2008.

[7] M. P. Deisenroth, D. Fox, and C. E. Rasmussen, "Gaussian processes for data-efficient learning in robotics and control", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 2, pp. 408–423, 2015.

[8] M. Deisenroth and C. E. Rasmussen, "Pilco: a model-based and data-efficient approach to policy search", in *Proc. of the 28th Int'l Conf. on Machine Learning (ICML-11)*, 2011.

[9] C. Devin, A. Gupta, T. Darrell, P. Abbeel, and S. Levine, "Learning modular neural network policies for multi-task and multi-robot transfer", *ArXiv preprint arXiv:1609.07088*, 2016.

[10] F. Farshidian, D. Pardo, and J. Buchli, "Sequential linear quadratic optimal control for nonlinear switched systems", *ArXiv preprint arXiv:1609.02198*, 2016.

[11] G. Favero and W. J. Runggaldier, "A robustness result for stochastic control", *Systems & Control Letters*, vol. 46, no. 2, pp. 91–97, 2002.

[12] J. Fu, S. Levine, and P. Abbeel, "One-shot learning of manipulation skills with online dynamics adaptation and neural network priors", *ArXiv preprint arXiv:1509.06841*, 2015.

[13] S. Gu, T. Lillicrap, I. Sutskever, and S. Levine, "Continuous deep q-learning with model-based acceleration", *ICML*, 2016.

[14] N. Heess, G. Wayne, D. Silver, T. Lillicrap, T. Erez, and Y. Tassa, "Learning continuous control policies by stochastic value gradients", in *NIPS*, 2015.

[15] G. Kahn, T. Zhang, S. Levine, and P. Abbeel, "Plato: Policy learning using adaptive trajectory optimization", *ArXiv preprint arXiv:1603.00622*, 2016.

[16] Z. Kalmár, C. Szepesvári, and A. Lörincz, "Module-based reinforcement learning: Experiments with a real robot", *Machine Learning*, vol. 31, no. 1, pp. 55–85, 1998.

[17] M. Kearns and S. Singh, "Near-optimal reinforcement learning in polynomial time", *Machine learning*, vol. 49, no. 2, pp. 209–232, 2002.

[18] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey", *The International Journal of Robotics Research*, p. 0 278 364 913 495 721, 2013.

[19] S. Levine and P. Abbeel, "Learning neural network policies with guided policy search under unknown dynamics", in *Advances in Neural Information Processing Systems*, 2014.

[20] 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.

[21] A. Mandlekar*, Y. Zhu*, A. Garg*, L. Fei-Fei, and S. Savarese (* equal contribution), "Adversarially robust policy learning through active construction of physically-plausible perturbations", in *IEEE Int'l Conf. on Intelligent Robots and Systems (IROS)*, 2017.

[22] D. Q. Mayne, E. C. Kerrigan, E. Van Wyk, and P. Falugi, "Tube-based robust nonlinear model predictive control", *International Journal of Robust and Nonlinear Control*, 2011.

[23] D. Mitrovic, S. Klanke, and S. Vijayakumar, "Adaptive optimal feedback control with learned internal dynamics models", in *From Motor Learning to Interaction Learning in Robots*, Springer, 2010, pp. 65–84.

[24] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.*, "Human-level control through deep reinforcement learning", *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[25] T. Moerland, J. Broekens, and C. Jonker, "Learning multimodal transition dynamics for model-based reinforcement learning", *ArXiv preprint arXiv:1705.00470*, 2017.

[26] I. Mordatch, K. Lowrey, G. Andrew, Z. Popovic, and E. V. Todorov, "Interactive control of diverse complex characters with neural networks", in *Advances in Neural Information Processing Systems*, 2015, pp. 3132–3140.

[27] I. Mordatch, K. Lowrey, and E. Todorov, "Ensemble-cio: full-body dynamic motion planning that transfers to physical humanoids", in *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, IEEE, 2015, pp. 5307–5314.

[28] A. Mueller, *How does the solution of a Markov decision process depend on the transition probabilities?* Inst. für Wirtschaftstheorie und Operations Research, 1996.

[29] M. Neunert, C. de Crousaz, F. Furrer, M. Kamel, F. Farshidian, R. Siegwart, and J. Buchli, "Fast nonlinear model predictive control for unified trajectory optimization and tracking", in *IEEE Int'l Conf. on Robotics and Automation (ICRA)*, 2016.

[30] L. Pinto, J. Davidson, R. Sukthankar, and A. Gupta, "Robust adversarial reinforcement learning", *ArXiv preprint arXiv:1703.02702*, 2017.

[31] A. Rajeswaran, S. Ghotra, S. Levine, and B. Ravindran, "EPOpt: learning robust neural network policies using model ensembles", *ArXiv preprint arXiv:1610.01283*, 2016.

[32] A. A. Rusu, N. C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, and R. Hadsell, "Progressive neural networks", *ArXiv preprint arXiv:1606.04671*, 2016.

[33] A. A. Rusu, M. Vecerik, T. Rothörl, N. Heess, R. Pascanu, and R. Hadsell, "Sim-to-real robot learning from pixels with progressive nets", *ArXiv preprint arXiv:1610.04286*, 2016.

[34] J. Schulman, S. Levine, P. Moritz, M. Jordan, and P. Abbeel, "Trust region policy optimization", *ICML*, 2015.

[35] S. Singh, A. Majumdar, J.-J. Slotine, and M. Pavone, "Robust online motion planning via contraction theory and convex optimization", in *Robotics and Automation (ICRA), 2017 IEEE International Conference on*.

[36] Y. Tassa, T. Erez, and E. Todorov, "Synthesis and stabilization of complex behaviors through online trajectory optimization", in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, IEEE, 2012, pp. 4906–4913.

[37] M. E. Taylor and P. Stone, "Transfer learning for reinforcement learning domains: A survey", *Journal of Machine Learning Research*, vol. 10, 2009.

[38] B. Thananjeyan, A. Garg, S. Krishnan, C. Chen, L. Miller, and K. Goldberg, "Multilateral surgical pattern cutting in 2d orthotropic gauze with deep reinforcement learning policies for tensioning", in *IEEE Int'l Conf. on Robotics and Automation (ICRA)*, 2017.

[39] E. Todorov and W. Li, "A generalized iterative lqg method for locally-optimal feedback control of constrained nonlinear stochastic systems", in *American Control Conference, 2005. Proceedings of the 2005*, IEEE, 2005, pp. 300–306.

[40] D. J. Webb and J. van den Berg, "Kinodynamic rrt*: asymptotically optimal motion planning for robots with linear dynamics", in *IEEE Int'l Conf. on Robotics and Automation (ICRA)*, 2013.

[41] Y. Zhu, R. Mottaghi, E. Kolve, J. J. Lim, A. Gupta, L. Fei-Fei, and A. Farhadi, "Target-driven visual navigation in indoor scenes using deep reinforcement learning", in *IEEE Int'l Conf. on Robotics and Automation (ICRA)*, 2017.