# Adapting Learned Robotics Behaviours through Policy Adjustment

Juan Camilo Gamboa Higuera, David Meger, and Gregory Dudek

***Abstract*— We present an approach to learning control policies for physical robots that achieves high efficiency by *adjusting* existing policies that have been learned on similar source systems, such as a similar robot with different physical parameters, or an approximate dynamics model simulator. This can be viewed as calibrating a policy learned on a source system, to match a desired behaviour in similar target systems. Our approach assumes that the trajectories described by the source robot are feasible on the target robot. By making this assumption, we only need to learn a mapping from the source robot state and action spaces to the target robot action space, which we call a *policy adjustment* model. We demonstrate our approach in simulation in the cart-pole balancing task and a two link double pendulum. We also validate our approach with a physical cart-pole system, where we adjust a learned policy under changes to the weight of the pole.**

## I. INTRODUCTION

This paper describes a method for quickly adapting learned controllers for a robot, so as to robustly handle changes in the dynamics that make the original controller fail. To avoid having to re-train the controller from scratch we take an approach based on the idea of *transfer learning*, which consists of reusing prior data and models to obtain savings in the computation and data requirements to learn a new task. Such savings are crucial for practical robotics systems. Every deployment to collect new data has a high cost in terms of human labour, and has a high risk of damaging the target robot system.

The process of transferring a *source* controller to optimize some desired behaviour involves experimentation with the *target* robot. Trials on the target provide insight that allows evaluation of differences from the source system and of the controller performance. A transfer method is responsible for utilizing these differences to adjust the controller appropriately. Adjusted controllers can be attempted again on the target, leading to new experience, further rounds of adjustment, and repetition until convergence. The goal of such a method is to use less interactions with the physical robot than *tabula rasa* learning, which is done without knowledge transfer from a source system. Transfer is attractive because methods that learn from scratch, such as PILCO [1] and Guided Policy Search (GPS) [2] which have been shown to enable powerful and flexible learned behaviours, have high a computational cost.

Some recent approaches to transfer [3], [4] use prior data to bias policy optimization in the target environment. While these methods are promising, since they are effective in reducing the number of interactions with the target robotic system, these methods have a computational cost comparable to learning from scratch.

To address the issues of high computational cost and minimizing the interactions with the physical system, we take inspiration from dataset aggregation approach for learning from demonstration [5], [6]. That is, we treat control learning (or transfer) as a supervised learning task, significantly reducing the computational complexity. We seek to imitate the optimal behaviour from the source system in the target system. Thus we build a dataset where the inputs are state-control pairs from trajectories obtained by applying the optimal controller in the source system, and the outputs are the controls in the target system that would reproduce such trajectories. Instead of searching for the target controls, we build an *inverse dynamics model* from data obtained in the target system. Using this dataset, we train a *policy adjustment* model that transforms the source controller into one that replicates the source trajectories in the target domain.

With such data driven models, we are making the assumption that the optimal behaviour in the source system is feasible in the target system. This is a non-trivial assumption that should be carefully considered. The target robot may not even be physically able to follow the trajectories from the source system due to physical limits. Even if the trajectories are feasible, they may describe a suboptimal behaviour in the target system. However, we believe that many practical scenarios do satisfy our conditions. We demonstrate our approach in scenarios when the conditions are true, achieving rapid and effective policy adjustments.

To further motivate our approach, consider the illustrative example of learning swimming controllers that stabilize an underwater robot in fresh water (e.g. as the scenario in [7]) and trying to apply the same controllers in salt water. As the density of the fluid changes, the hydrodynamics will be different enough for the original controllers to fail in performing the task. Different forces are required to counteract different effective buoyancy. However, the ideal stable path described by the robot in the source environment (fresh water in our example) contain information that can be exploited to speed up the learning on the target environment. A similar situation can be found for flying robots that slightly bend a wing or ground robots that puncture a tire.

We describe a method for training such policy adjustment model, using data from the source sytem and an inverse dynamics model of the target system to generate a dataset for supervised learning. We demonstrate our approach in simulation in the cart-pole balancing task and a two link

---

The authors are part of the Center for Intelligent Machines and the School of Computer Science, McGill University, Montreal, Canada. This work was supported by NSERC through funding for the NSERC Canadian Field Robotics Network {gamboa,dmeger,dudek}@cim.mcgill.ca

double pendulum. We also validate our approach with a physical cart-pole system, where we adjust a learned policy under changes to the weight of the pole. We show how this method requires drastically less computation than *tabula rasa* learning, and discuss its limitations and future improvements.

## II. BACKGROUND

### A. *Learning motor behaviours*

The method proposed in this work requires an algorithm that optimizes motor controllers on a source robot system. Several algorithms have been proposed in the literature, using techniques based on stochastic optimization, dynamic programming or heuristic search. In the experiments reported on this paper, we used Probabilistic Learning in Control (PILCO) [1] to train the source controllers, a method which works by building a model of the system dynamics to perform policy optimization, while minimizing the amount of interaction with the physical system.

Our work is mostly related to methods that use guiding examples to train the behaviors using supervised learning. The Dataset Aggregation DAGGER algorithm [5] takes as an input a control policy provided by expert demonstrations and uses supervised learning to train a policy that mimics the expert. Then, the policy is iteratively updated by executing the policy on the target system and adding the new experience data to the dataset for training the policy. This is similar to the algorithm described below, except in our case the expert demonstrations come from a source environment with different dynamics to those of the target environment.

Guided Policy Search [2] uses a similar idea, except that it uses a trajectory optimization algorithm to generate the example trajectories. In their work, the authors use Differential Dynamic Programming (DDP) to obtain a locally optimal trajectory that performs the desired behaviour. Based on this solution, the authors construct a model for sampling trajectories that will be used for estimating the gradient of the control policy and to estimate the performance of the control policy. The procedure is run iteratively, aggregating the sample trajectories at each iteration in a similar way as [5]. The resulting algorithm also has a strong similarity to our work, except our formulation is based on the transfer learning paradigm [8].

### B. *Transfer Learning*

Recent work [3], [4] showed how data from low fidelity simulators can be used to bias policy optimization in the target environment. The data transferred from low fidelity simulators includes value functions, state transition models, learned policies, and experience data. This is done in order to reduce the number of interactions with the target robotic system. In these approaches, the learning algorithm for the source and target environments is the same. Only the initial conditions (e.g., the prior mean predicting robot motions) differ due to the transferred data. While the authors have shown less trials are required on the target system, the computational cost of learning is still roughly as high as when learning from scratch, which can be a significant burden for complex systems.

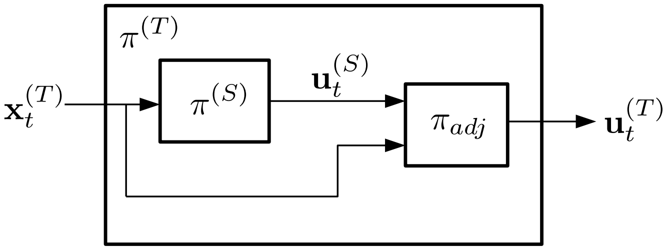


Fig. 1. The controller used in the target robot system $T$, consists of the original source policy $\pi^{(S)}$ and the *policy adjustment* model.

Cully *et al.* [9] demonstrated a system for gait adaptation that relies on equipping a robot with a simulator model and a library of pretrained behaviours. This system fits a *transferability score function* that attains high values when the physical robot experience agrees with the simulated behaviour, and low values when there is a discrepancy. The authors demonstrated how this system could be used to adapt gaits on a hexapod robot when deployed on a physical robot with a variety of electrical and mechanical failures. This is similar to our work in the sense that the target system data is used to modify a controller trained in a different setting. The difference is that they use the data to select a previously trained controller from a library, while we use the data to adjust a previously trained controller.

As mentioned in the previous sections, we do not continue the learning process with the same algorithm, since this might be quite computationally expensive. Instead, we make a powerful assumption allowing a much more efficient approach, and avoid repeating the expensive computations done to obtain the source policies.

## III. ADAPTING LEARNED POLICIES THROUGH POLICY ADJUSTMENT

Our goal is to restore performance of a robotic system performing a task after a change occurs that impacts its dynamics; making its original controller fail at performing the task. We aim to do so by using previous learning experience as a guide. Ideally, we would like to find a new controller that optimizes the same objective as the original one, e.g. to find an optimal policy under the new dynamics.

We propose an approach consisting of computing a *policy adjustment* model $\pi_{adj}$; which maps control policies $\pi^{(S)}$, from the source environment $S$, to a new policy $\pi^{(T)}$, in the target domain $T$, that reproduces the original behaviour. We define the policy adjustment model to have the following form

$$\pi_{adj} : \mathcal{X} \times \mathcal{U} \rightarrow \mathcal{U} \tag{1}$$

For our proposed method, we make the following assumptions about the source $S$ and target $T$ environments.

- Tasks in both $S$ and $T$ can be represented as discrete time Markov Decision Processes (MDP): $(\mathcal{X}, \mathcal{U}, \mathcal{T}^{(S)}, c)$ and $(\mathcal{X}, \mathcal{U}, \mathcal{T}^{(T)}, c)$.

- The state and action spaces of $S$ and $T$ correspond to the description of the same robotic system.
- Although the state transition dynamics $\mathcal{T}^{(S)}$ and $\mathcal{T}^{(T)}$ may differ, any sequence of states in $S$ drawn according to the optimal policy $\pi^{(S)}$ has non-negligible probability in $T$, under some policy $\pi^{(T)}$. In other words, the optimal behaviour of the robot in $S$ is physically plausible in $T$.
- The sampling period for the discrete time MDP is small enough so that the one timestep inverse kinematics of $T$ can be approximated by a bijective function.

Note that, although the ideal solution would be to find a new controller that optimizes the accumulated costs (obtained by evaluating $c$), reproducing the original behaviour under the new dynamics might be suboptimal. Figure 1 illustrates how we integrate the adjustment model in the target robot's controller. At every time step $t$, the target controller $\pi^{(T)}$ takes as input the robot's current state $\mathbf{x}^{(T)}$ and computes the action $\mathbf{u}^{(S)}$ that would have been optimal in $S$. The adjustment model will modify this action in order to reproduce the state transitions observed in the source data, producing the target control output $\mathbf{u}_t^{(T)}$. Note that the policy adjustment model may be trained with different source policies, as both the action and the state are inputs to the adjustment. In our experiments, we let the adjustment model fall back to the original controller in the regions of the state space where there is no data, as explained in the following subsections.

Our method consists of the following steps

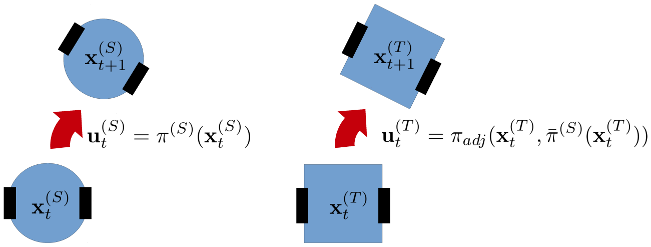
- We sample trajectories using the learned controller to be used as examples of the desired behaviour in $T$.
- Next we fit an *inverse dynamics* model from trajectories sampled in $T$.
- Finally we use the inverse dynamics and the example trajectories to generate a dataset for learning the policy adjustment using supervised learning.

Figure 2 provides an illustration of our method. The last two steps are performed iteratively in order to gather useful data to train the policy adjustment model. We provide a more detailed explanation of the steps of our method in the following sections, and summarize our method in Algorithm 1.

### A. Source Learning and Trajectory Sampling

Given a desired task specification for a robotic system, in the form of a target state and a cost function, we use any existing algorithm that produces a control policy $\pi^{(S)}$ that minimizes the cost associated to the task. In our experiments, we use the PILCO to obtain source policies in the form of radial basis functions (RBF) networks.

Given $\pi^{(S)}$, we sample a set of example trajectories $D^{(S)}$ from $S$. By sampling we mean picking slightly different starting locations for the system, and introducing noise in the case of using a deterministic simulation. In the case of a physical system we skip the noise injection and rely on the non-deterministic behaviour of the physical system. These sampled trajectories are stored as state-action-state triples, $(\mathbf{x}_t^{(S)}, \mathbf{u}_t^{(S)}, \mathbf{x}_{t+1}^{(S)})$, which we will use to guide the desired behaviour in $T$; i.e. we aim to imitate the behaviour described by this set of trajectories.


$\mathbf{x}_{t+1}^{(S)}$
$\mathbf{u}_t^{(S)} = \pi^{(S)}(\mathbf{x}_t^{(S)})$
$\mathbf{x}_t^{(S)}$
$\mathbf{x}_{t+1}^{(T)}$
$\mathbf{u}_t^{(T)} = \pi_{adj}(\mathbf{x}_t^{(T)}, \bar{\pi}^{(S)}(\mathbf{x}_t^{(T)}))$
$\mathbf{x}_t^{(T)}$

(a) Source domain (b) Target domain

Fig. 2. The goal of the policy adjustment $\pi_{adj}$ is to replicate trajectories from the source domain in the target domain. We do this by generating a dataset $(\mathbf{x}_t^{(S)}, \mathbf{u}_t^{(S)}, \mathbf{u}_*^{(T)})$, where $\mathbf{u}_*^{(T)} = g^{(T)}(\mathbf{x}_t^{(S)}, \mathbf{x}_{t+1}^{(S)})$, and training $\pi_{adj}$ by supervised learning; e.g. GP regression

### B. Target Trajectory Sampling and Learning Inverse Dynamics

Increasing our knowledge of the target system requires choosing exploratory actions to gather new data, which is a canonically difficult problem. Two standard approaches to sample trajectories in $T$ are (1) random behaviour (to explore); or (2) by applying $\pi^{(S)}$ (to exploit). The expected benefits of these two simple approaches will depend on how related $S$ and $T$ are. If $S$ and $T$ are the same, the obvious choice is to use the policy directly to obtain the desired behaviour as rapidly as possible. On the other hand, as the difference between $S$ and $T$ becomes more pronounced, it will become harder to find the actions that replicate the trajectories from $S$ in $T$. As real systems fall across a mix of these scenarios, we attempt to intelligently accomplish both factors. We apply the source policy while encouraging exploration by adding a small amount of noise to the applied controls. Specifically, we select this noise proportional to the confidence of our policy adjustment model, as will be explained later. The sampled trajectories are stored as a dataset of state-action-state triples $D^{(T)}$.

From this dataset, we obtain a representation of the inverse dynamics in $T$,

$$g : \mathcal{X}^{(T)} \times \mathcal{X}^{(T)} \rightarrow \mathcal{U}^{(T)} \tag{2}$$

which maps pairs of states to the action that may cause the transition. We currently assume that $g$ is locally bijective, i.e. there is only one action that explains any given state transition, which we have found reasonable for a small enough time discretisation period. This allows us to use any data-driven approach to fit an inverse dynamics model instead of performing a costly search, or deriving an exact inverse dynamics model. We found that using a Gaussian Process (GP) Regression for fitting $g$ allows us to filter predictions of inverse dynamics based on confidence levels.

### C. Training the Policy Adjustment Model

The policy adjustment model produces a new policy for $T$, $\pi_{adj}$, by transforming the output of $\pi^{(S)}$ into an action that replicates the behaviour from $S$ in $T$, as illustrated in Figure 2.

To obtain $\pi_{adj}$, we take a state-action-state triplet $(\mathbf{x}_t^{(S)}, \mathbf{u}_t^{(S)}, \mathbf{x}_{t+1}^{(S)})$ from the example trajectories in $D^{(S)}$ and use the inverse dynamics model $g$ to obtain the action $\mathbf{u}_t^{(T)}$ that is most likely to produce the same transition in $T$.

For every input pair $(\mathbf{x}_t^{(S)}, \mathbf{x}_{t+1}^{(S)})$ we obtain $\mathbf{u}_t^{(T)}, \Sigma_{\mathbf{u}_t}^{(T)}$ from the inverse dynamics model $g$; i.e. the mean prediction and its covariance, which is diagonal in the case $g$ is a GP model with deterministic inputs. We use $\Sigma_{\mathbf{u}_t}^{(T)}$ to filter out samples with low confidence. We repeat this procedure for all the example trajectories from $S$ and form a new dataset $D^{(adj)}$ consisting of triplets $(\mathbf{x}_t^{(S)}, \mathbf{u}_t^{(S)}, \mathbf{u}_t^{(T)})$ for training $\pi_{adj}$.

We chose to use GP regression to train $\pi_{adj}$ from $D^{(adj)}$. In our implementation, we set the prior mean of the GP for $\pi_{adj}$ to be zero, and train the adjustment model on triplets $(\mathbf{x}_t^{(S)}, \mathbf{u}_t^{(S)}, \mathbf{u}_t^{(T)} - \mathbf{u}_t^{(S)})$; i.e. the output is an additive change we need to apply to the source controls. At test time, we compute the controls to apply at a state $\mathbf{x}$ as $\mathbf{u} = \pi^{(S)}(\mathbf{x}) + \pi_{adj}(\mathbf{x}, \pi^{(S)}(\mathbf{x}))$. This allows the target controller to fall back to the source controller in regions where it has no data, as the prior prediction from $\pi_{adj}$ is zero with high variance. Using a GP model allows us to implement the previously mentioned exploration strategy for sampling trajectories in the target system. When executing the controller $\pi^{(T)}$, we will sample $\mathbf{u}_t^{(T)}$ from the output normal distribution given by the mean and covariance of the prediction from $\pi^{(T)}$. Since we filtered out the predictions from the dynamics model with low confidence, the variances in the predictions from $\pi_{adj}$ will be large in regions where we have no data, which will result in samples that are considerably different from the mean, leading to more aggressive exploration.

Our method continues by sampling trajectories from the target domain, fitting the inverse dynamics and training the policy adjustment model multiple times, each time gathering more data to train the policy adjustment model. We summarize our method in Algorithm 1.

## IV. EXPERIMENTAL RESULTS

### A. Adjusting Policies in Simulation

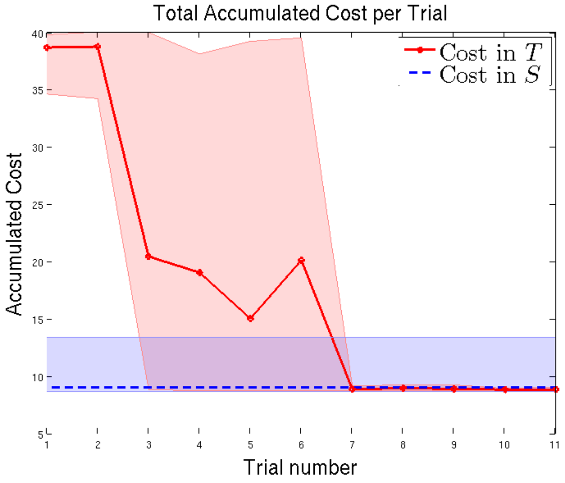
We have tested our approach in the cart-pole and inverted double pendulum tasks. In both cases the task is to balance the pendulum in an upright position. The source tasks were learned by training Radial Basis Function (RBF) policies via PILCO [1]. For the cart-pole experiment, each trial has a time horizon of 4.0 seconds, the time step for controlling the system is 0.1 seconds, and the controller is composed of 10 RBF units. For the double pendulum experiment, the time horizon is 3.0 seconds, the time step is 0.1 seconds, and the controller is composed of 100 RBF units. The cart-pole task took 5 iterations to converge, while the double pendulum task took 10 iterations.

**Algorithm 1:** Policy Adjustment

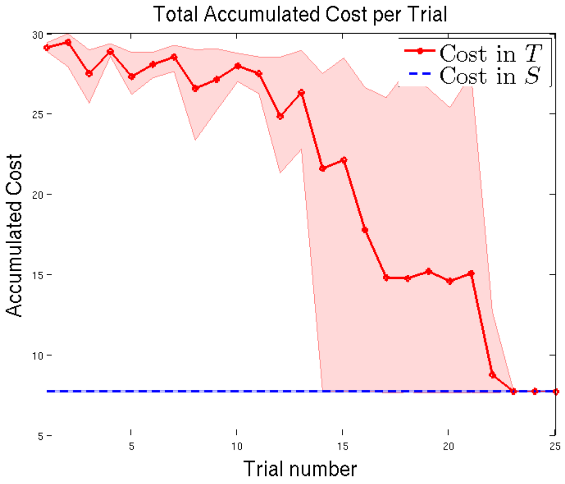
```
Inputs: Source MDP $S : (\mathcal{X}, \mathcal{U}, \mathcal{T}^{(S)}, c)$
        Target MDP $T : (\mathcal{X}, \mathcal{U}, \mathcal{T}^{(T)}, c)$
        Policy Learning algorithm $\mathbf{L}_1$
        Supervised learning algorithm $\mathbf{L}_2$
        Supervised learning algorithm $\mathbf{L}_3$
        The maximum number of target trials $N_T$
Output: A policy adjustment model $\pi_{adj}$
$\pi^{(S)} \leftarrow \mathbf{L}_1 (\mathcal{X}, \mathcal{U}, \mathcal{T}^{(S)}, c)$
Sample $D^{(S)}$ applying $\pi^{(S)}$ in $S$
$D^{(T)} \leftarrow \emptyset$
$i \leftarrow 0$
$\pi_{adj} \leftarrow \pi^{(S)}$
while $i < N_T$ do
    $D^{(adj)} \leftarrow \emptyset$
    Sample $D_i^{(T)}$ by applying $\pi_{adj}$ in $T$
    $D^{(T)} \leftarrow D^{(T)} \cup D_i^{(T)}$
    $g \leftarrow \mathbf{L}_2 (D^{(T)})$
    for each $(\mathbf{x}_t^{(S)}, \mathbf{u}_t^{(S)}, \mathbf{x}_{t+1}^{(S)}) \in D^{(S)}$ do
        $\mathbf{u}_t^{(T)} \leftarrow g (\mathbf{x}_t^{(S)}, \mathbf{x}_{t+1}^{(S)})$
        $D^{(adj)} \leftarrow D^{(adj)} \cup (\mathbf{x}_t^{(S)}, \mathbf{u}_t^{(S)}, \mathbf{u}_t^{(T)})$
    end for
    $\pi_{adj} \leftarrow \mathbf{L}_3(D^{(adj)})$
end while
return $\pi_{adj}$
```

The target systems shared the morphology of the source, but had balanced links with twice the mass of the original. This change is sufficient that the source policies produce very poor performance, and adjustment is a pressing need; however, the the change is not so large as to break our method's assumptions. We performed numerous repetitions of both learning the original source policy and also performing the adjustment to the heavier target system in order to verify the repeatability of our approach. For each learning repetition of adjustment, the only inputs are the policy and sample trajectory from the source system. Our method is given no knowledge at all about what has changed in the target, and must interact with it over a sequence of episodes to determine the appropriate adjustment (the simulator is automatically reset to a start state to begin each episode).

The results of these simulated trials, displayed in Figure 3, illustrate the effectiveness of our transfer approach on this task. The dashed blue lines represent the mean accumulated cost over 50 different runs in the source environment. The solid red lines represent the variability in performance per learning iteration. The thinner solid lines are the minimum and maximum values over the 50 runs. Our method was able to match the original performance without having to learn from scratch, in 3-7 iterations (12-28 seconds of experience) for the cartpole task, and 15-20 iterations (45-60 seconds of experience) for the double pendulum task. Both of these results represent a fraction of the computational cost, as we



(a) Cart-pole.



(b) Inverted double pendulum.

Fig. 3. Results of our policy adjustment method in two simulated domains.

will now examine.

Timing results from these experiments show that our method is considerably faster than learning from scratch. Each of the cart-pole learning iterations in PILCO (learning a dynamics model plus doing a policy search iteration) takes, on average, 6 minutes on an machine with a Intel(R) Core(TM) i7-6700HQ CPU @ 2.60GHz. In contrast, our method takes 5-10 seconds for training the inverse dynamics and the adjustment model. The computational cost of each iteration of our method is dominated by the training of the inverse dynamics and policy adjustment functions, for which efficient approximation methods exist [10], [11]. For comparison, a single learning iteration with PILCO requires training a dynamics model (of similar complexity as our method), and evaluating the gradient of the policy over the sequential controls and dynamics rolled out over the entire episode length, which is the most computationally intensive part. Our method does not require computing policy gradients or roll-outs, requiring much less computation time. Note however that our algorithm decreases the computation time, but not necessarily the hardware interaction time.

### *B. Adjusting Policies on a Physical Cart-Pole System*

In order to verify the effectiveness of our method on real robot hardware and to avoid simulation bias, we constructed a single-motor cart-pole system illustrated in Figure 4. We verified that *tabula rasa* learning with PILCO was able to perform balancing on this original system, which has a pole weight of 45 g, as well as several modified systems, where a range of weights (27.5 g, 45 g, 90 g and 180 g) were attached roughly two thirds of the way towards the end of the pole. This lead to effective masses of 1.5x, 2x, 3x and 5x compared with the original system. For each case PILCO learning resulted in effective balancing, but required around 7 attempts with the real hardware interspersed with learning, which amounted to roughly 45 minutes of overall elapsed time.

We then conducted a set of policy adjustment experiments, always taking the learned policy from the lightest system (the one with no added weight) as the source input, while the cart-pole was weighed down by the additional masses. Figure 5 displays one sequence of state-space trajectories followed by our system during policy adjustment for the pole with 2x the original mass. Based on learning from one execution of the source policy in the target to gather data, *Adjusted Traj. 1* is the first path followed by our system, and while it is improved, it still falls short of balancing, with the pole quickly falling and dragging the cart far to the positive X direction. This performance yields additional fruitful data for our method to exploit, as can be seen from *Adjusted Traj. 2*, which manages to pass through the target balancing point, however with too great a velocity, leading to the pole over-rotating. With this additional data and another 10 seconds of computation, the next learned policy produces *Adjusted Traj. 3*, which produces stable balancing, following a close approximation to the *SRC in Source* trajectory which was considered the optimal performance learned by PILCO on the source system.

Figure 6 summarizes the outcome of applying the proposed *policy adjustment* algorithm to each of the 4 added-mass target scenarios. The left column shows the progression of cumulative cost per episode, where quick convergence to low or zero cost represents effective learning. The middle and right columns show single dimensions of the state space, with stabilization at the dotted blue lines indicating successful balancing has been achieved. The accompanying video shows the experiments that were run in order to obtain these results.

It is evident that learning from scratch with PILCO on the source system was effective, as is our adjustment method when applied to added weights of up to 3x. In the 5x added weight scenario, our method did not converge to successful balancing. We observed that the motor torques being attempted by the adjusted policies in this case saturated our actuator, so it is likely that this dynamic system is not able to follow the guiding trajectories from the source that our method attempts to match. Solving this heaviest scenario may require an additional swing back-and-forth, which our method will never attempt. We note that, while

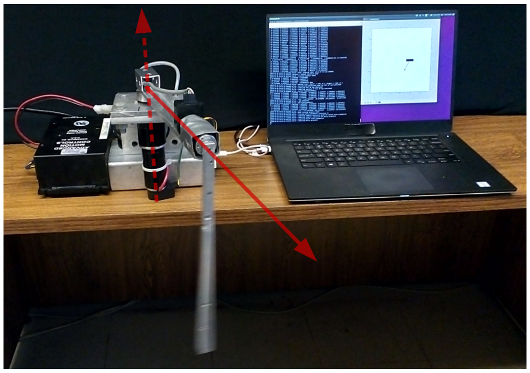


Fig. 4. Our experimental setup for the cart-pole domain. The solid arrow denotes the rotation axis of the pendulum. The dashed line denotes the rotation axis of the actuator.

the adjustment failed, this can be easily corrected, as the cost progression shows a lack of convergence, so a hybrid system could initiate the learning of a new policy from scratch if desired.

## V. CONCLUSIONS

We have described a policy adjustment approach capable of learning to perform tasks on difficult dynamical systems in only seconds through the use of an existing policy and example trajectories from a similar system as a strong guide to learning. Our method is based on two primary components: (1) experimentation with the target system's dynamics to produce an efficient data-driven inverse dynamics model using a Gaussian Process; and (2) guided-sampling using trajectories that were known to be effective in the source system to build up a dataset of samples that inform a supervised learning policy optimization. The combined method has been successful in learning on cart-pole and double pendulum tasks in simulation as well as on a physical cart-pole robot. In all cases, the mass of the balanced links can be doubled or more, completely invalidating the effectiveness of the original learned policy, and our approach is able to restore performance in just a handful of trials and typically a minute of elapsed time. To our knowledge, ours is the fastest adaptation algorithm that has been demonstrated to date for this particular task.

While the demonstrations in this paper have been focused on technical efficacy, it is interesting to consider the potential of our method for use on-board practical fielded robotic systems. The reader will recall that our method depends on an effective policy for the source environment. If this is learned from scratch, it will, of course, be expensive. However, our approach allows for paying the price of an expensive computation only once. As the robot's dynamics change (such as picking up tools or wearing a tire), applying the policy adjustment algorithm is quick and easy. This reveals that our approach is well-suited to being a background optimization process on board robots that interact with difficult environments and are likely to experience minor failures or wear-and-tear that results in subtle, but not catastrophic changes in dynamics (e.g., swimming robots with flexible flippers, tire wear, manipulators that become slightly fouled).

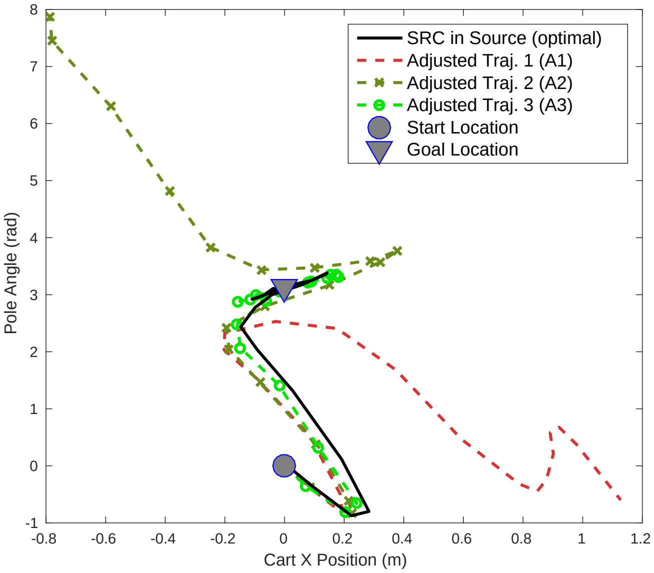


Fig. 5. The sequence of trajectories performed by our system for a physical cart-pole system with an additional 45 g weight added near its end, effectively doubling the pole mass to be balanced. Our adjustment algorithm sequentially improves the target behaviour to match the optimal trajectory known for the source system, rapidly converging to strong performance in only three trials.

One of the main outstanding issues that we hope to investigate in the future is the selection of which data points to include in the dataset used for supervised training of $\pi_{adj}$. Selecting starting states from the source domain $\mathbf{x}_t^{(S)}$ is desirable as it defines the trajectory that we want to follow, but it has the problem that we may not have enough data in $D^{(T)}$ to compute the inverse dynamics $g^{(T)}$ for such states. On the other hand, selecting starting states from the target domain $\mathbf{x}_t^{(T)}$ is desirable because this corresponds to regions of the state space where we can compute $g^{(T)}$. However these states might have not been visited when training the source policy, and would not help bringing the robot closer to the optimal behaviour. Our current approach of weighting samples by their uncertainty has proven the most effective amongst those we attempted, but it will be interesting to continue additional techniques in future work.

Unlike the prior work on using a simulator in RL [12], [3], we are not continuing to optimize the policy in the target domain, using the source policy to seed the optimization. Instead, we use our assumption to synthesize a control policy through the policy adjustment model $\pi_{adj}$, which should work for multiple tasks. Our approach has strong similarities with the Dataset Aggregation DAGGER algorithm [5], except that we have no expert that could demonstrate the trajectories, other than the policy trained on a *different domain* (perhaps a simulator). It will be promising to pursue methods that can appropriately utilize guidance from multiple sources, such as a human expert as well as a policy that has been learned from a similar domain, and we believe our efficient approach is well-suited for the real-time requirements that would be presented by such an integrated system.

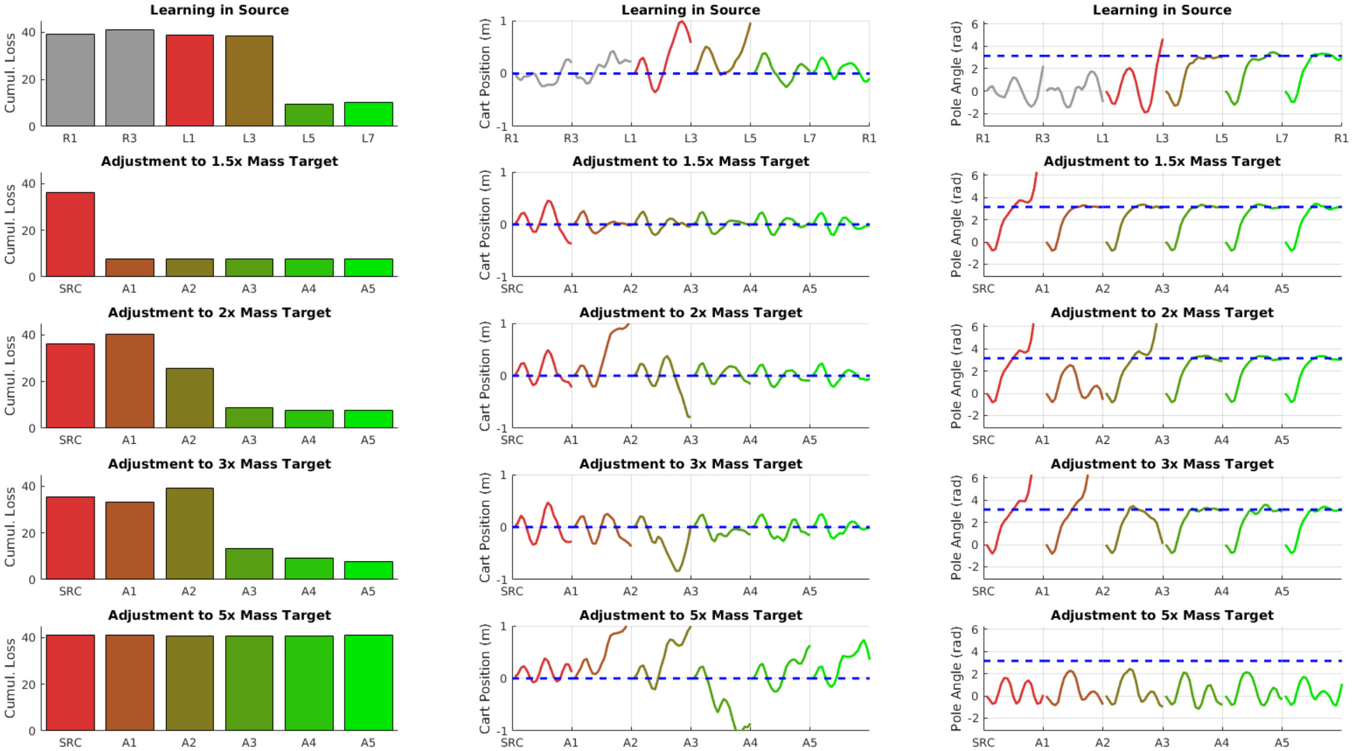


Fig. 6. Results for numerous configurations of the physical cart-pole system. The first row shows learning from scratch with a simple pole (no weights added), which includes 4 random trials (R1 and R3 shown) and 7 iterations of expensive policy-gradient optimization (L1, L3, L5 and L7 shown), in 45 minutes elapsed time. Rows 2 through 5 show adjustment to successively heavier added weights. Our method first executes the policy from the source (SRC) to begin collecting data and then runs 5 trials with adjusted policies (A1 through A5). Up to the 3x heavier configuration, our method learned to perform the task as well as in the source system; by trial A3 and in less than 1 minute elapsed time. Our method was unsuccessful in adjusting to the 5x heavier system, likely because our motor torque limits prevented matching the source trajectory.

## ACKNOWLEDGMENT

We would like to acknowledge the NSERC Canadian Field Robotics Network (NCFRN) for its funding support. We would also like to thank Anqi Xu for the useful discussions and script used for Figure 6, and Nikhil Kakodkar for the helpful comments.

## REFERENCES

[1] M. Deisenroth, D. Fox, and C. Rasmussen, "Gaussian processes for data-efficient learning in robotics and control," *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, 2014.

[2] S. Levine and V. Koltun, "Guided policy search," in *Proceedings of The 30th International Conference on Machine Learning*, 2013.

[3] M. Cutler and J. P. How, "Efficient reinforcement learning for robots using informative simulated priors," in *IEEE International Conference on Robotics and Automation (ICRA)*. Seattle, WA: IEEE, May 2015.

[4] M. Cutler, T. J. Walsh, and J. P. How, "Reinforcement learning with multi-fidelity simulators," in *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA'14)*, 2014.

[5] S. Ross, G. J. Gordon, and J. A. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," *arXiv preprint arXiv:1011.0686*, 2010.

[6] S. Ross and J. A. Bagnell, "Agnostic system identification for model-based reinforcement learning," *arXiv preprint arXiv:1203.1007*, 2012.

[7] D. Meger, J. C. Gamboa Higuera, A. Xu, P. Giguère, , and G. Dudek, "Learning legged swimming gaits from experience," in *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2015.

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

[9] A. Cully and J.-B. Mouret, "Behavioral repertoire learning in robotics," in *Proceedings of the 15th annual conference on Genetic and evolutionary computation*. ACM, 2013, pp. 175–182.

[10] M. Lázaro-Gredilla, J. Quiñonero-Candela, C. E. Rasmussen, and A. R. Figueiras-Vidal, "Sparse spectrum gaussian process regression," *The Journal of Machine Learning Research*, 2010.

[11] Y. Gal and R. Turner, "Improving the gaussian process sparse spectrum approximation by representing uncertainty in frequency inputs," in *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, 2015.

[12] J. Kober and J. Peters, "Reinforcement learning in robotics: A survey," in *Reinforcement Learning*. Springer, 2012, pp. 579–610.