Sample-efficient Imitation from Observation on a Robotic Arm

Sean Geiger
The University of Texas at Austin
May 2019

Abstract

Recent applications of generative adversarial networks to the task of imitation learning have achieved significant performance improvements over other methods for imitating complex behaviors. However, these networks require many demonstration examples and learning iterations to produce a policy that is successful at imitating a demonstrator’s behavior. Meanwhile, in the field of reinforcement learning, linear quadratic regulators (LQR’s) have been used to achieve sample-efficient learning in techniques that employ neural networks such as guided policy search [10]. In this paper, we investigate whether using LQR’s in a similar way with generative adversarial networks can also increase sample efficiency in imitation learning. We propose an algorithm to combine the two techniques and conduct experiments using an imitation task on a robot arm and in simulation to compare the learning rate of generative adversarial learning with our mixed approach. We find that this new method results in successful imitation learning with fewer samples than generative adversarial imitation networks alone.
## Contents

1 Introduction 3

2 Related Work 4
   2.1 Imitation Learning 4
   2.2 Guided Policy Search and Linear Quadratic Regulators 5

3 Background 6
   3.1 Generative Adversarial Imitation Learning 6
   3.2 Guided Policy Search and Linear Quadratic Regulators 7
      3.2.1 LQR with Unknown Dynamics 8
      3.2.2 GPS: Generalizing LQR with Nonlinear Policies 8

4 Approach 9

5 Experiments 11
   5.1 Setup 11
   5.2 Experimental Design 12
      5.2.1 Comparison to GAIFO 14
      5.2.2 Generalization 14
      5.2.3 Performance on Physical Arm 16

6 Discussion 16

7 Conclusion and Future Work 18
1 Introduction

Teaching new actions to robot actors through demonstration is one of the most attractive methods for behavior learning. While robots can learn new behaviors using reinforcement learning with a pre-specified reward function [18], significant exploration is often required to extract the behavior from the reward. In some cases, denser reward functions can help speed up the exploration process, but this approach requires a certain level of skill and understanding of the reinforcement learning process, and can often result in unexpected behaviors when the reward function doesn’t precisely guide the action. Instead, teaching a robot a behavior simply by demonstrating it removes the requirement of explicitly specifying a reward function altogether. Anyone who knows how to perform the task can demonstrate it without understanding the learning process, and the learning process requires much less exploration. This process—learning from demonstration (LfD)—aims to take a series of observed states (e.g. joint angles, position in space) and actions (e.g. decisions to move a joint at some speed) and extract a policy that approximates the demonstrated behavior [1].

While being able to imitate a behavior after observing the state and actions of a demonstrator is useful, there are many situations where the actions of the demonstrator are unknown. Common approaches to LfD require both the states and actions of the demonstrator to be recorded [1]. In imitation from external observation (IfO), on the other hand, just the observable states of the demonstrator are known—no action information is available. Imitating behaviors solely from observable data greatly expands the set of possible demonstrators: behaviors could be learned from in-person human demonstrators or even the vast collection of videos available online.

While imitation from external observation has been studied and performed with some success for two decades [3], recent advances in deep neural networks have widened the set of behaviors that can be imitated and the ways that demonstration data can be collected. One way deep learning has been applied to IfO is through generative adversarial networks [19, 5]. In this approach—generative adversarial imitation from observation (GAIfO)—one network learns a control policy for imitating the demonstrator while the other learns to discriminate between the demonstrator’s behavior and that of the imitator. While GAIfO advanced the state of the art in imitation from observation, it comes with its own set of challenges. While early IfO attempts that used simpler regressed models required a small amount of training data [13], deep networks are notorious for requiring orders of magnitude more training data, and GAIfO is no exception. Some of the possible benefits of the applications of IfO break down when a high sample size is required. In practice, GAIfO has been largely limited to being studied in simulation. In simulation, many experiences and large demonstration sets can be collected quickly. Physical demonstrations are more costly to perform, and real-time constraints limit the speed at which control policies can be evaluated and thus behavior learned. The sample size required to train the adversarial networks places a limit on the practicality of collecting trajectories in real-time on a real robot. For imitation from observation to work
on a physical robot, a higher degree of sample efficiency is required.

Deep reinforcement learning has faced similar obstacles with learning with limited samples, especially in the context of robotic control policies with complex dynamics. Guided policy search (GPS) has become a powerful method for sample-efficient learning in which simple models that are trained for trajectory optimization direct policy learning \([10, 8, 11, 9]\). GPS achieves this sample efficiency in part by gaining insight into dynamics through the iterative training of linear quadratic regulators (LQR's) on a set of trajectory controllers.

Motivated by the sample efficiency of LQR in the setting of reinforcement learning, we seek to answer the question: will using the techniques of linear quadratic regulators in combination with generative adversarial imitation from observation provide a more sample-efficient approach to imitation learning? To investigate this question, we propose an algorithm that combines GAIfo and LQR and hypothesize that this technique will allow an imitator to approximate a demonstrator’s actions in fewer iterations than GAIfo alone. We apply the proposed algorithm to a 6-degree-of-freedom robot arm to learn to imitate behaviors from a set of low-level state trajectories. The experimental results confirm our hypothesis: we find that this new method results in successful imitation learning with fewer samples than generative adversarial imitation networks alone.

In Section 2 of this paper, we discuss previous work related to this topic. In Section 3, we cover the techniques involved in GAIfo and LQR. Section 4 describes our approach to combining LQR and GAIfo into one functional algorithm. In Section 5, we share our experimental setup and results, and we discuss results in Section 6. Finally, in Section 7, we summarize and discuss potential future work.

2 Related Work

Our approach to sample-efficient imitation learning is built upon previous works in the field of learning from demonstration as well as policy learning in the field of reinforcement learning.

2.1 Imitation Learning

In a survey of the topic of learning from demonstration, Argall et al. subdivide imitation learning techniques into those that learn from external observation (referred to in this paper as Imitation from Observation) and those that learn from observation directly from the teacher [1]. In the former case, the actions of the teacher are unknown, so only a time-ordered series of states are available for policy extraction. In addition to the type of demonstration data available to the imitator, techniques for imitation learning also differ in the way they approach the problem. Two popular approaches to imitation learning have been behavioral cloning [14] and inverse reinforcement learning [13, 15]. Behavioral cloning views state-action transitions as a supervised learning problem, while
inverse reinforcement learning works to find a cost function under which the expert demonstrator is optimal.

In 2015, Ho and Ermon used generative adversarial networks to imitate policies when both states and actions are available using a technique called generative adversarial imitation learning (GAIL) [5]. One imitator network attempts to imitate the policy while another attempts to discriminate between the imitation and provided demonstration data [4]. Both networks are randomly initialized, and on each iteration, the policy of the imitator is executed, the discriminator is updated based on a loss function against the demonstration data, then the imitator is updated based on the loss of the discriminator.

While GAIL has access to the demonstrator’s actions, generative adversarial learning from observation (GAIfO) addresses imitation learning in situations where actions are unknown [19]. More specifically, GAIL characterizes the cost function as a function of states and actions while the cost function in GAIfO is instead characterized by state transitions. GAIfO provides two implementations: one in that uses low-level state data (e.g., direct measurements of robot arm joint angles) to form a policy and another that handles imitation from visual data by adding convolutional layers at the front of the imitator and discriminator that take multiple frames of video as input. While both of these techniques have been shown to discover accurate imitation policies, to date, they have only been evaluated in a simulated experimental domain. Because experiments consist of thousands of iterations in which each iteration includes executing a policy several times, the time required for monitoring experiments in the real world, for example on real robots, is prohibitive. In this work, our goal is to resolve GAIfO’s sample inefficiency to the extent that it can be applied to learning a behavior on a real robot.

2.2 Guided Policy Search and Linear Quadratic Regulators

In reinforcement learning–policy learning through environment-provided reward functions only—direct policy search in a large state-action space requires numerous samples and often can fall into poor local optima. Guided policy search (GPS) is a method to improve the sample efficiency of direct policy search and guide learning in a large space away from poor local optima [10]. The basis of GPS is to use trajectory optimization to focus policy learning on high-reward actions.

In guided policy search under unknown dynamics, time-varying linear Gaussian models of the dynamics for a small set of specific tasks are first trained to fit a small set of sample data through LQR [8]. These Gaussian controllers are then sampled to generate samples to optimize a general policy for a model with thousands of parameters that would typically require much more training data. Specifically, samples in regions of trajectories that have been found to lead to higher reward are generated, guiding the policy learning.

GPS has had success in learning policies in reinforcement learning situations with complex dynamics and high-dimensional inputs, including training a policy
that directly controls the torque on motors in a robot arm to perform a task like screwing a cap on a bottle solely from raw images of the system [9]. Current applications of GPS have focused on reinforcement learning, and the technique’s applications to imitation learning have not been adequately explored. The work presented in this paper aims to do just that.

3 Background

Our approach to sample-efficient imitation learning is based on work in two areas: generative adversarial imitation learning, which uses a dual-network approach to fitting a policy to expert data, and linear quadratic regulators, a sample-efficient approach to extracting a linear policy from a quadratic cost function. In this section, we review both of these techniques in detail.

3.1 Generative Adversarial Imitation Learning

Generative adversarial learning consists of training two networks that operate in tandem: a generative network $G$, that attempts to generate data similar to training data, and a discriminative network $D$, that learns to differentiate the generated data and the training data [4]. $G$ is trained through backpropagation to increase the likelihood of $D$ making a mistake. At the unique optimal solution, $G$ perfectly fits the training data distribution, and $D$ cannot differentiate between the output of $G$ and the training data.

Generative adversarial imitation learning [5], data is recorded as the expert demonstrates a behavior, resulting in a set of example trajectories $\tau_E$ that represent the expert policy $\pi_E$. $G$ is a network that learns a policy $\pi_I$ that generates trajectories $\tau_I$ similar to $\tau_E$. Each trajectory is a set of state-action transitions $\{s_0, a_0, ..., s_n, a_n\}$. The discriminator network is parameterized by weights $w$ and maps input trajectories to a score between 0 and 1: $D : S \times A \rightarrow [0, 1]$, with an output value of 1 denoting that the input trajectory matches the distribution of expert (demonstration) trajectories $\tau_E$, and an output of 0 meaning the two distributions are separate. The generative network is a policy parameterized by weights $\theta$ that chooses actions when provided with states in order to produce a new trajectory. In a reinforcement learning view, the discriminator network provides a cost function that changes $\theta$ to move the distribution of trajectories created by $\pi_I$ towards the distribution of trajectories generated by $\pi_E$.

In each iteration of GAIL, a number of trajectories $\tau_I$ are sampled from the current imitation policy $\pi_I$. These trajectories are used to approximate the gradients of their respective policies with respect to the output of $D$, with the approximate expected value denoted as $\mathbb{E}_\tau$. The weights $w$ of the discriminator network are then updated according to the Adam [6] technique for optimization of objective functions along the gradient, which is given by:

$$\mathbb{E}_{\tau_I} [\nabla_w \log (D_w (\tau))] + \mathbb{E}_{\tau_E} [\nabla_w \log (1 - D_w (\tau))]$$  (1)
Following the update to $D$, the weights $\theta$ of the generative policy are updated using the technique of Trust Region Policy Optimization [16] under the cost function $\log(D_w(\tau))$ where $D_w$ is the newly updated discriminator network. The full algorithm is summarized in Algorithm 1.

<table>
<thead>
<tr>
<th>Collect expert demonstrations: $\tau_E$;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Randomly initialize: $G(\theta), D(w)$;</td>
</tr>
<tr>
<td>while $\pi_I$ not converged do</td>
</tr>
<tr>
<td>Collect trajectories $\tau_I$ from current policy $\pi_I$ using $G(\theta)$;</td>
</tr>
<tr>
<td>Update $w$ using a step of Adam optimization according to gradient defined in (1);</td>
</tr>
<tr>
<td>Update $\theta$ in a bounded step of TRPO using the cost function with new $w$: $\log(D_w(\tau))$;</td>
</tr>
<tr>
<td>end</td>
</tr>
</tbody>
</table>

Algorithm 1: Generative adversarial imitation learning

In generative adversarial imitation from observation [19], actions are not available in the demonstration data and are not present in collected trajectories. The update steps remain the same, but the trajectories now only consist of states: $\tau = \{s_0, s_1, ..., s_n\}$.

3.2 Guided Policy Search and Linear Quadratic Regulators

Linear quadratic regulators (LQR’s) learn control policies under two assumptions [2]:

1. The dynamics of the environment are linear. This means that the transition from a particular state given an action $f(s_t, a_t)$ can be represented as the product of the state/action and a matrix $F_t$ plus a constant vector $f_t$:

$$f(s_t, a_t) = F_t \begin{bmatrix} s_t \\ a_t \end{bmatrix} + f_t$$

2. The cost is quadratic. The cost is represented by a quadratic term $C_t$ and a linear vector $c_t$:

$$c(s_t, a_t) = \frac{1}{2} \begin{bmatrix} s_t \\ a_t \end{bmatrix}^T C_t \begin{bmatrix} s_t \\ a_t \end{bmatrix} + \begin{bmatrix} s_t \\ a_t \end{bmatrix}^T c_t$$

The optimal action at a given time can be found by minimizing the quadratic cost at each previous timestep. In situations where the dynamics are assumed to be close to linear but are not completely known or are nondeterministic, the linear transition function is often replaced by a conditional probability specified
under a normal Gaussian distribution, with a mean of the linear dynamics and a covariance:

\[ p(s_{t+1}|s_t, a_t) = \mathcal{N}(F_t \left[ \begin{array}{c} s_t \\ a_t \end{array} \right] + f_t, \sigma^2) \]

When the covariance is constant (independent of the state and action), the optimal policy is identical to the non-stochastic LQR.

In non-linear systems where the cost is not quadratic, the techniques of LQR can be used by approximating the dynamics with a first-order Taylor expansion and approximating the cost with a second-order Taylor expansion:

\[ F_t = \nabla_{s_t, a_t} f(s_t, a_t), \quad C_t = \nabla_{s_t, a_t}^2 c(s_t, a_t), \quad c_t = \nabla_{s_t, a_t} c(s_t, a_t) \]

Iterative linear quadratic regulators (iLQR’s) can be used to find optimal policies under non-linear models by running LQR with the approximated dynamics, then updating the dynamics fit on each iteration [12].

### 3.2.1 LQR with Unknown Dynamics

LQR assumes that the dynamics of the environment are known. Learning dynamics for a given situation involves building a model to define \( f(s_t, a_t) \) from a set of observed state/action transitions \( D = ((s_t, a_t, s_{t+1})) \). A simple approach to this model building is to use linear regression to estimate the dynamics, finding some matrices \( X \) and \( Y \) that model the transition as \( f(s_t, a_t) = X s_t + Y a_t + c \), or in a stochastic environment, \( p(s_{t+1}|s_t, a_t) = \mathcal{N}(X s_t + Y a_t + c, \sigma^2) \). Modelling dynamics with a Gaussian approximation of the linear regression (often called linear Gaussian models) has the advantage of being very sample-efficient.

To avoid the erroneous pursuit of an incorrect global optimal, a set of local models can be used to replace a global model. The most expressive case of local models is a set of models with a single model for every timestep. In the linear regression approach, this amounts to fitting new \( X_t \) and \( Y_t \) for every timestep, often called time-varying controllers. Because dynamics are often highly correlated between timestep, this approach can be refined by using a global model as a prior for a Bayesian linear regression at each timestep [9].

Because linear regression can overshoot optimals of non-linear dynamics, policy adjustment can be bounded so that each iteration’s update to the model’s transition distribution (or trajectory distribution) is not too large. This can be achieved with a bound on the Kullback–Leibler (KL) divergence—a relative measure of divergence between distribution—between the previous trajectory distribution and the current trajectory distribution.

### 3.2.2 GPS: Generalizing LQR with Nonlinear Policies

Fitting time-varying linear gaussian controllers to observed dynamics and choosing a policy based on these controllers using LQR can perform well in situations where the policy and dynamics are similar to those used to create the controllers. Because the Gaussian controllers are so sample efficient, this technique
also tends to achieve faster improvements than model-free policies. However, policy performance can be bounded by the non-conformance of the true dynamics to its linear approximation and the cost to its quadratic approximation, as well as the simplicity of the linear Gaussian controllers. These linear Gaussian controllers also do not generalize well to situations with similar dynamics but different trajectories.

Guided policy search attempts to combine the performance of a nonlinear policy like a deep neural network with the sample efficiency of iLQR. In GPS, a number of linear gaussian controllers \( \{p\} \) are trained through iLQR to model a set of different trajectories \( \{\tau\} \). These trajectories have conditions that vary in the same dimension that the policy is expected to generalize in. For example, a policy that navigates to a point in space from any starting position would require training controllers for a number of different starting positions. Then, once these controllers have converged, they are used to train a complex nonlinear policy \( \pi_\theta \) to imitate \( \{\tau\} \). The intuition behind this process is that the dynamics of the environment that are learned quickly in the gaussian controllers can be conveyed to the nonlinear model to give it a significant head-start in the optimization process.

4 Approach

For LQR to be useful in an imitation learning scenario, it can no longer depend on a pre-specified reward function that defines the task. Instead, the trajectory optimization step in LQR should be based on the existing controller’s ability to imitate the expert demonstration. To achieve this effect, we train a discriminator network on each iteration and use an approximate version of its loss on the sampled trajectories to optimize the controllers.

Given a set of expert demonstration trajectories \( \{\tau_E\} \), our algorithm begins by randomly initializing a time-varying model \( p \) to model the trajectory dynamics. \( p \) is specified as \( p(s_{t+1}|s_t, a_t) = \mathcal{N}(F_t \begin{bmatrix} s_t \\ a_t \end{bmatrix} + f_t, \sigma^2) \). Given a set of sample trajectories \( \{\tau_s\} \), \( F_t, f_t, \) and \( \sigma^2 \) are fit to the sample data at each timestep using bayesian linear regression with a Normal-inverse-Wishart prior. For this prior, we fit the entire trajectory sample to a Gaussian Mixture Models (GMM), which previous research with GPS has found to be effective for this prior [9]. On every iteration of the algorithm, the current controller is used to collect \( \{\tau_s\} \), which is then used to refit the dynamics.

Following the dynamics update, the discriminator is updated using a bounded Adam update according to the gradient specified in (1). For LQR to be able to use the discriminator as a cost function, it needs a quadratic interpretation of the cost. First, the discriminator—a function of state transitions—is combined with the Gaussian dynamics models—functions of states and actions—to create a composite cost function \( C(s_t, a_t) \). This composite function is quadratically approximated by taking the first and second order Taylor expansions of the cost:
Figure 1: The GAIfo + LQR Algorithm
Finally, an iteration of LQR uses this cost approximation \(c_q\) to optimize the trajectory to form a new linear Gaussian controller. The step size of this update is bounded by the KL-Divergence compared to the previous iteration. The main components of this approach are depicted in Figure 1.

5 Experiments

To evaluate the performance of LQR + GAIFO, we studied the algorithm’s ability to imitate a simple reaching task on a robot arm—both on a physical arm and in a simulator.

5.1 Setup

For a testing platform, we used a Universal Robotics UR5, a 6-degree-of-freedom robotic arm. We used an implementation of LQR provided in the GPS implementation by the authors [7] along with a publicly-available integration for the UR5 arm. The task that is demonstrated is a simple reaching task in which the arm begins in a consistent, retracted position and reaches towards a point in Cartesian space. When the end effector (the gripper at the end of the arm) reaches this point, the arm stops moving. This task is shown in Figure 3. Expert demonstrations are recorded by training the LQR reinforcement learning policy to move towards a goal position until the policy converges and the goal position is consistently reached. This policy is then executed and recorded a number of times to create the demonstration data. For equipment safety purposes, only three of the six joints move during both demonstration and training, effectively converting the arm to 3-degrees-of-freedom. This locks the arm into a single plane of movement (parallel to the whiteboard shown in Figure 2a), limiting the possibility for collisions during training. While leaving joints unused simplifies the state space and the dynamics of the arm, we expect the results to generalize to higher degrees-of-freedom.

We modified the software to record the state of the arm and the action chosen at every timestep of the GPS trajectory execution. For the initial experiments, the state—a 12 dimensional vector—consisted of:

1. Joint angles (3)
2. Joint velocities (3)
3. Cartesian distance to the goal position from the end effector (3)
4. Cartesian velocity of the end effector (3)
The action at each timestep consists of a vector of goal joint angles for each joint. At each timestep (a timestep lasts 0.1 seconds), the arm receives a command to move to a goal position. If the arm has not reached the goal angles by the next timestep, the arm redirects towards the new goal positions. For testing in simulation, we used the Gazebo simulation environment (Figure 2b) with a model of the UR5. Each trial lasts for 100 timesteps (10 seconds) and ends regardless of the end effector reaching the goal state. At each iteration, the policy being evaluated is executed five times to collect five sample trajectories. The policy is also evaluated once without noise per iteration, and the performance according to the cost function is logged.

The cost function used takes into account the distance from the end effector to the target position, weighted linearly as the trial progresses. With the distance from the goal position to the end effector at a given timestep $d_t$, the cost of a trajectory with $n$ timesteps is calculated as:

$$C(\tau) = d_{t_n} + \sum_{i=0}^{n-1} \frac{i}{n} d_{t_i}$$

The same cost function is used to train the expert through reinforcement learning as well as to evaluate the performance of the imitator. In this sense, the task of imitation learning can be seen as recovering the cost function that guided the expert [19]. For a more complex task or more specific cost function than the one studied, it’s possible that the imitator could recover the task behavior correctly while not performing well in the eyes of the cost function, or vice versa. However, for the arm reaching task, the cost function is simple and directly related to the task, making it appropriate as an evaluator of imitation performance. For the imitation tasks, this cost function was used to evaluate each trajectory sample at a given iteration. The results were normalized on a range from zero to one, with zero mapping to the average cost of a random policy, and one mapping to the cost achieved by the discriminator. A policy that performs as well as the discriminator would achieve a score of one on this normalized performance scale.

To evaluate the performance of the GAIfO-only approach, code made available privately from the author of [19] was instrumented to interface with the arm control and simulation platform. Trials for the GAIfO-only approach also involved taking five samples per iteration, in the same way as the LQR+GAIfO approach. The GAIfO generator network was updated using Proximal Policy Optimization, which has been shown to be more sample efficient than the Trust Region Policy Optimization approach initially used for GAIL [17].

5.2 Experimental Design

We conducted three main experiments to evaluate LQR+GAIfO. In the first experiment, the learning rate is compared to learning under GAIfO alone. In the second experiment, we test LQR+GAIfO’s ability to generalize to unseen target points. Finally, we compare the performance of the algorithm in the simulated environment to the physical arm.
The UR5 Robot Arm
The UR5 Arm Modeled in the Gazebo Simulator

Figure 2: The Experimental Domains

Target Point

Starting Position → Moving Toward Target → Stop at Target

Figure 3: A depiction of the reaching task being demonstrated in the simulator. The arm starts in a retracted position and reaches the end effector toward the target point, stopping when the target point is reached.
Figure 4: Learning rate comparison of LQR+GAIfO to GAIfO in simulation. The normalized performance is shown, with 0.0 denoting the performance of a random policy, and 1.0 denoting the performance of the demonstrator. The error bars show the mean standard error of the policy samples.

5.2.1 Comparison to GAIfO

To compare the learning rate of LQR+GAIfO to that of GAIfO alone, we ran trials for both algorithms for 100 iterations and tracked the policy’s performance at each iteration using the cost function described in Section 5.1. This process was repeated for both algorithms (n=30 for LQR+GAIfO, n=55 for GAIfO) to collect average performance data. The algorithms’ performance along with the mean standard error is plotted in Figure 4. The performance of LQR+GAIfO quickly exceeds GAIfO alone and peaks around iteration 30. Eventually, LQR+GAIfO’s policy performance starts to degrade around iteration 60 and is passed in performance by GAIfO around iteration 90.

5.2.2 Generalization

To test LQR+GAIfO’s ability to generalize a policy for a point that is not in the expert demonstration data, we collected expert demonstration trajectories for 8 points on the edge of a square (shown in Figure 6). For each point, we trained the expert and recorded five sample trajectories when the expert converged. Then, after choosing a subset of the points on the square as \( \{ \tau_E \} \), we tasked the arm with moving to a point in the center of the square. Because the center point was not in \( \{ \tau_E \} \), the control policy was required to generalize
Figure 5: The learning rate of LQR+GAIF in simulation when tasked with reaching an unseen point given a demonstration set with a varying number of demonstration points.
Figure 6: Points collected for expert data in the generalization experiment. A varying number of points were chosen from the edges of a square surrounding the target point. Demonstrated trajectories to these chosen points form the expert demonstration set.

5.2.3 Performance on Physical Arm

The LQR+GAIfO algorithm was run on both the simulator and the physical arm to examine how closely simulated performance mapped to real-world performance. Over 25 iterations, the policy performance on the physical arm began to surpass the performance of the simulated arm, as shown in Figure 8.

6 Discussion

Our research began by asking if a combination of LQR and GAIfO could increase sample efficiency in imitation learning. The comparison of LQR+GAIfO to GAIfO suggests that LQR+GAIfO can indeed produce a policy that is better at imitating a behavior in a limited number of iterations, confirming our hypothesis. The steep initial learning curve of LQR+GAIfO indicates significantly higher sample efficiency compared to GAIfO alone. However, the performance of LQR+GAIfO seems to degrade consistently around iteration 60. Without this performance regression, LQR+GAIfO would outperform GAIfO past iteration 100. The reason for this degradation is unknown, and further research is needed.
Figure 7: The normalized performance of LQR+GAfO in simulation compared to the normalized performance of LQR+GAfO on the physical UR5 arm over 25 iterations.
to understand its cause. In addition, even without this degradation, the GAIfO approach would eventually surpass the performance of LQR+GAIfO, likely due to the ability of the generator network in GAIfO to produce more complex policies than those that can be represented with linear Gaussian controllers in LQR.

Although most of the ability for a policy to perform a task that is different from the expert trajectories in GAIfO and GPS result from a complex generator function, the linear Gaussian controllers in LQR+GAIfO still have the ability to generalize to some degree. As expected, the ability to successfully generalize increases with demonstration trajectories, as shown in Figure 5. It is possible that the performance increase as the number of demonstration points increased was caused by a higher cardinality of \( \{\tau_E\} \), not the variation of trajectories within \( \{\tau_E\} \). A future experiment that maintains the cardinality of the expert trajectory set across trials could help answer this question. Future work integrating the full GPS approach would likely lead to better generalization.

We studied the performance of LQR+GAIfO on the physical arm to validate the tractability of this technique on a real robot and to establish a sense of how directly the performance studied in the simulator would translate to the physical arm. Our results, as seen in Figure 7, show that the policy performance seen in the simulator can be trusted to model policy performance on the real arm. Surprisingly, the performance of LQR+GAIfO on the physical arm exceeds the simulator performance. While the reason for this result is unknown, it is possible that the noise introduced by the physical arm as a result of actuator noise or other physical effects lead to wider exploration and faster policy improvement. If this is the case, it could be possible to achieve similar performance in the simulator by introducing more policy noise. Further experimentation is required to fully understand this effect.

7 Conclusion and Future Work

We have found that combining generative adversarial imitation from observation with Linear Quadratic Regulators leads to faster learning of imitation behavior over fewer samples than with GAIfO alone, confirming our hypothesis. While LQR+GAIfO doesn’t reach the absolute imitation performance of GAIfO over an extended training period with thousands of samples, achieving adequate imitation performance with limited samples opens the door to imitation research on physical robotic systems, for which imitation learning has posed logistical challenges in the past.

While LQR is a powerful technique by itself, a policy based solely on Gaussian controllers has limits in complexity. Work in GPS has already produced a method for combining sample-efficient Gaussian controllers with a deep network model that is trained through the controllers. Using a deep network as part of the policy offers increased performance in the long run and greatly increased generalization ability. Incorporating this deep network policy driven by importance-weighted samples of the linear Gaussian controllers is an obvious
and promising next step for this work.

To validate the LQR+GAIfO technique, we represented the expert trajectories using low-level data like the cartesian position of the arm’s end effector. GAIfO has had success in using higher level data—like a visual recording of the demonstrator—as the state in trajectories. Additionally, GPS has used auto-encoded visual data to successfully train linear gaussian controllers [9]. Pursuing imitation learning from visual data alone would greatly widen the situations in which demonstration data could be collected. Adding a convolutional layer to the discriminator so that it can accept visual data is a natural next step for extending this research.
References


